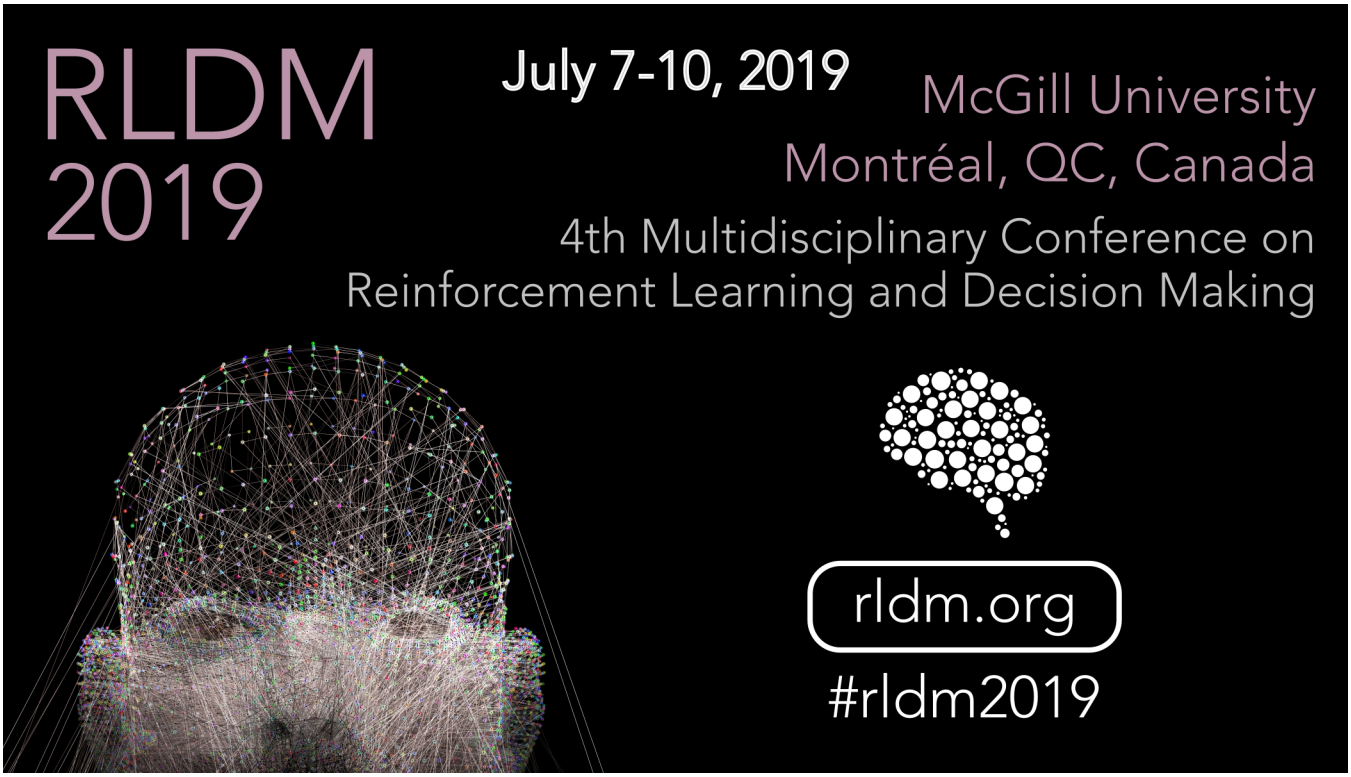


Extended Abstracts

www.rldm.org

TABLE OF CONTENTS

Modeling cooperative and competitive decision-making in the Tiger Task

Saurabh Kumar
Institute of Systems Neuroscience
University Medical Center Hamburg, Germany
s.kumar@uke.de

Tessa Rusch
Institute of Systems Neuroscience
University Medical Center Hamburg, Germany
t.rusch@uke.de

Prashant Doshi
Department of Computer Science
University of Georgia, GA, USA
pdoshi@cs.uga.edu

Michael Spezio
Psychology & Neuroscience
Scripps College, CA, USA;
Institute of Systems Neuroscience
University Medical Center
Hamburg, Germany
mspezio@scrippscollege.edu

Jan Gläscher
Institute of Systems Neuroscience
University Medical Center
Hamburg, Germany
glaescher@uke.de

Abstract

The mathematical models underlying reinforcement learning help us understand how agents navigate the world and maximize future reward. Partially observable Markov Decision Processes (POMDPs) – an extension of classic RL – allow for action planning in uncertain environments. In this study we set out to investigate human decision-making under these circumstances in the context of cooperation and competition using the iconic Tiger Task (TT) in single-player and cooperative and competitive multi-player versions. The task mimics the setting of a game show, in which the participant has to choose between two doors hiding either a tiger (-100 points) or a treasure (+10 points) or taking a probabilistic hint about the tiger location (-1 point). In addition to the probabilistic location hints, the multi-player TT also includes probabilistic information about the other player's actions. POMDPs have been successfully used in simulations of the single-player TT. A critical feature are the beliefs (probability distributions) about current position in the state space. However, here we leverage *interactive POMDPs* (I-POMDPs) for the modeling choice data from the cooperative and competitive multi-player TT. I-POMDPs construct a model of the other player's beliefs, which are incorporated into the own valuation process. We demonstrate using hierarchical logistic regression modeling that the cooperative context elicits better choices and more accurate predictions of the other player's actions. Furthermore, we show that participants generate Bayesian beliefs to guide their actions. Critically, including the social information in the belief updating improves model performance underlining that participants use this information in their belief computations. In the next step we will use I-POMDPs that explicitly model other players as an intentional agents to investigate the generation of mental models and Theory of Mind in cooperative and competitive decision-making in humans.

Keywords: Theory of Mind, Tiger-task, Cooperation, Competition, Bayesian modeling, I-POMDP

Acknowledgements

J. G. was supported by the Bernstein Award for Computational Neuroscience (BMBF 01GQ1006) and J.G. and M.S. were supported by a Collaborative Research in Computational Neuroscience (CRCNS) grant (BMBF 01GQ1603; NSF 1608278). T.R. was supported by a PhD scholarship from the German National Merit Foundation.

Extended Abstract

1 Introduction

Reinforcement learning (RL) has its roots in artificial intelligence, control theory, operation research and has proven to be a powerful framework for cognitive neuroscience decision-making under uncertainty. *Markov decision processes* (MDPs) - the mathematical model underlying RL - help robots to pursue the goal of maximized total future reward by guiding their decisions when the state space is fully known. The real world, however, is imperfect with noisy observations and unexpected environment changes, where the current state of the world is often uncertain. The *partially observable Markov decision processes* (POMDPs) extend MDPs for situations of state uncertainty by proposing a belief distribution over possible states and using Bayesian belief updating for estimating this belief distribution in each moment (Kaelbling, Littman and Cassandra 1998).

The iconic Tiger Task played a crucial role in developing this computational framework by providing a test bed for simulating decision-making of a single agent in an uncertain world. The task mimics the setting of a game-show, in which the agent is presented with two doors, one of which hides a tiger (incurring a large loss) and the other one hides a pot of gold (incurring small win). The POMDP framework has been subsequently extended for multi-agent settings resulting in *interactive partially observable Markov decision process* (I-POMDP) (Gmytrasiewicz und Doshi 2005), in which two or more agent interact in an uncertain world. A crucial element of this framework is that agent build models of the other players and use them to predict others' choices and make better decisions themselves. The Tiger Task was again used in initial simulations of agents an their mental contents in this interactive setting.

Given the crucial role of the Tiger Task in formulating POMDPs and I-POMDPs it is surprising that little empirical data exist on this this task. Here, we set out to fill this gap by collecting choice data from human participants engaging in the single- and multi-agent Tiger Task, the latter being the focus of this paper. Furthermore, following Doshi (Doshi 2005) we devised a cooperative and a competitive version of the multi-agent Tiger Task and exposed two groups of subjects to them. In a series of model-free and model-based analyses of behavioural choice patterns we demonstrate a cooperative context elicits better choices and accurate predictions of the other player's actions and that subjects generate Bayesian beliefs to guide their actions. Critically, including the social information from the other player in the belief updating improves model performance, which underlines that participants pay attention to the other player and use it in formulating beliefs about the state of the world.

2 Task and hypothesis

The goal of the Tiger Task is to maximize the reward by opening the door hiding the gold (+10 point) and to avoid opening the door with the tiger (-100 points). In each step there are 3 actions available to the participant: open left door (OL), open right door (OR), or listen (L), which results in a probabilistic hint about the location of the tiger (growl left (GL), or growl right (GR)), but also costs 1 point. Thus, participant can accumulate evidence about the tiger location through repeated L actions. After each open action the position of the tiger is reset randomly to one of the two doors (tiger left (TL) or tiger right (TR)).

In the multi-player version the participants receive an additional probabilistic hint about the actions of the other player: creak left, or creak right (indicating that the other player might have opened one of the doors), or silence (S) indicating that the other player probably listened. Creaks suggest that the location of the tiger might have reset and that currently accumulated beliefs about the tiger location are void. Opening the door reveals the correct location of the tiger and the participant get the associated reward with additional knowledge of the tiger reset. In our implementation of the Tiger Task participant were also asked to predict the other player's actions at each step before choosing their own action (see Figure 1A for task sequence).

The competitive and cooperative versions differ in the structure of the payoff matrix: while the cooperative version incentivizes concurrent open actions by both players (see Figure 1C bold marking), the competitive version provides the maximum reward, if the correct door hiding the gold is opened, while the other player opens the wrong door hiding the tiger (see Figure 1B bold marking). Comparing the two versions,

we expected that participants will take more hints to come to a consensus in cooperative context to avoid confusing the other player and generate a more predictable behavior. We also expected more identical actions and more accurate predictions of the other player's actions during cooperation.

3 Results

We invited 58 participants (30 cooperate, 28 compete) to play the multi-player version of the game. In the model-free analysis we observed that the participants in the cooperative context took more hints than in the competitive context. In addition, prediction accuracy was higher during cooperation. These outcomes were both in line with our expectations. Participants in the competitive version exhibited fewer identical actions when compared to cooperation (Figure 2A-C).

Participants in the Tiger Task form beliefs about the states of the game (TL or TR) based on the probabilistic hints (GL or GR) and – in the multi-agent Tiger Task – the information from the other player (CR or CL). Because there are 3 distinct actions (OL, OR, L) available, we decided to model the action $a(t)$ at each step t as an ordered logistic regression model: $a(t) = ß_0 + ß_1 * b(t)$, where $b(t)$ is the belief about the location of the tiger.

The Tiger Task has only 2 states (TL and TR), which implies a unidimensional belief distribution with both states at the end of the range of possible beliefs. This belief distribution is updated on every step with the observations following the current action. We compared two version of belief updating: a simple "beta-belief" model, which uses the mode of a beta distribution as the point estimate of the belief and is updated by adjusting the parameters of the beta distribution with the observations (the probabilistic hints following L actions). The second model is a Bayesian belief updating model with take the previous belief as the prior and calculates the likelihood based on the observation and transition function. We also tested two versions of the Bayesian updating model (*Eq 1*) without and E*q 2*) with the inclusion of the social information (also see Figure 3A-B as an example):

$$b(t) = \frac{p(gc)*b(t-1)}{p(gc)*b(t-1)+(1-p(gc))*(1-b(t-1))} \quad (1)$$

Where, p(gc) is the probability of the hint being correct and b(t-1) is the previous belief about the tiger location.

$$b(t) = \frac{p(cc)*p(oo)}{p(cc)*p(oo)+(1-p(cc))*(1-p(oo))} * p(reset) + 1 - \frac{p(cc)*p(oo)}{p(cc)*p(oo)+(1-p(cc))*(1-p(oo))} * b(t-1) \quad (2)$$

Where, p(cc) is the probability of the hint about the partners' action being correct and p(oo) is the probability of the partner opening the door, while reset is the probability of the tiger being placed after a door is opened (0.5 for a random placement).

Models were estimated using the Stan software package that implements and hierarchical Bayesian workflow. Formal model comparison using LOOIC (Leave-one-out information criterion) revealed that the Bayesian belief update model resulted in a better fit than the beta-belief model (LOOIC (Bayesian belief) = 5107.75, LOOIC (Beta belief) = 8530.70). In control analysis, we expanded the set of predictors in the ordered-logistic model with additional task variables like the number of hints taken, previous outcome and an interaction between them (Model 2-5), but found the simpler model with just the belief as a predictor (Model 1) outperforms these more comprehensive predictor sets (Figure 4A-B). Furthermore, we compared the Bayesian belief update without the social information (Eq 1) to the update with the social information added (Eq 2) and concluded that the social information adds a significant improvement in the model prediction (see the scales of LOOIC values in Figure 4A and 4B).

4 Outlook

We used an ordered logistic discrete choice model with Bayesian belief updating and demonstrated that including the social information is providing a much better model fit to the data. This suggests that participants in the multi-agent Tiger Task do incorporate the information from the other player into their valuation process. However, our Bayesian belief model falls short of an important feature that is likely shaping strategic social decisions: it treats the information from the other players as just another piece of information from the environment and not as an intentional agent that processes the information in a similar way.

I-POMDPs are a computational framework that explicitly computes the beliefs of the other player as an intentional agent as part of the model of the first player. Thus, it is an ideal framework for modeling Theory of Mind of another player in a quantitative way (his goals, intentions, and beliefs). Following our Bayesian belief model, we will also model the Tiger Task within the I-POMDP framework and compare belief computations of the other player in the competitive and cooperative version of the task.

5 References

Doshi, Prashant. „Optimal sequential planning in partially observable multiagent settings." Ph.D. dissertation, University of Illinois at Chicago, 2005.

Gmytrasiewicz, Piotr J., und Prashant Doshi. „A framework for sequential planning in multi-agent settings." *Journal of Artificial Intelligence Research*, 2005.

Kaelbling, Leslie Pack, Michael L. Littman, und Anthony R. Cassandra. „Planning and acting in partially observable stochastic domains." *Artificial Intelligence*, 1998.

6 Figures

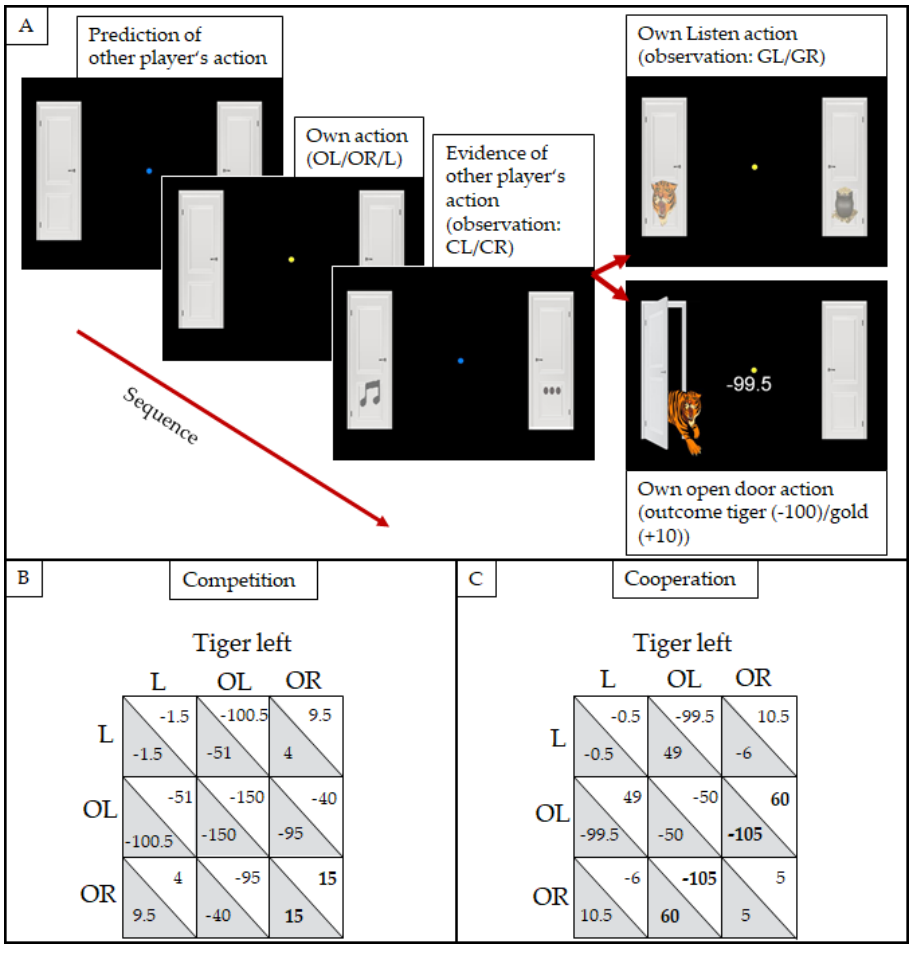


Figure 1: (A) An example of sequence for the multi-player version of the task. The player predicts the action of the other player (indicated by the blue fixation dot) followed by the players own action choice (indicted by the yellow fixation dot). This is followed by the probabilistic evidence about the other player's action (CL/CR). The next screen is either the probabilistic hint about the tiger location (GL/GR if L was chosen) or the door is opened (for OL or OR actions) revealing the tiger location. (B) The joint payout matrix in the competitive context is shown for the tiger being on the left side. The bold numbers show the best and worst choice indicating that the best own outcome is achieved if the correct door (with the gold) is opened, while the other player open the wrong door (with the tiger). (C) The joint payout matrix in the cooperative context is shown when the tiger is on

the left side. The bold numbers showing the best choice indicating that the maximum payoff is achieved, when both players open the correct door at the same time.

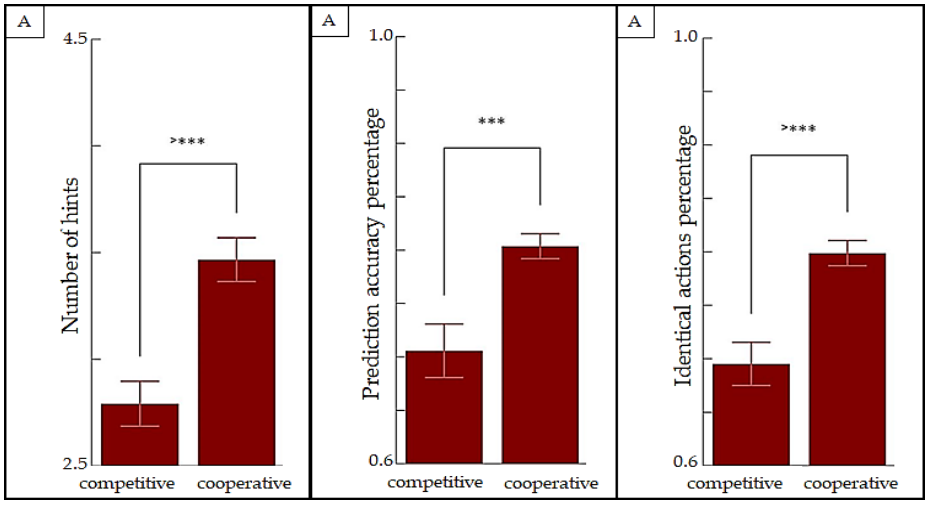


Figure 2: (A) In the multi-player version of the task the participants significantly took more hints in the cooperative context when compared to the competitive context. Participants also had significantly higher prediction accuracy and identical actions (showing coordination) in the cooperative context compared to the competitive one in (B) and (C) respectively.

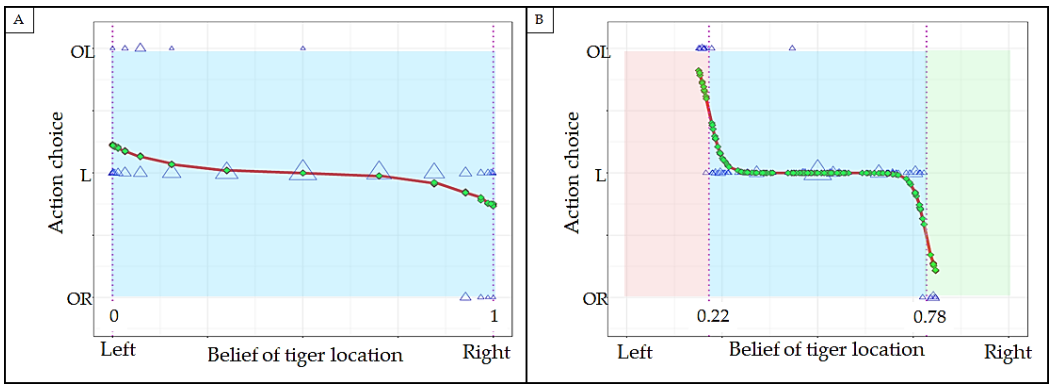


Figure 3: (A) An example model behavior of the multi-player version of the tiger task without the social information (see Eq 1) is shown here. The bold red line is the model prediction while the blue triangles are the actual participant action choices given their computed beliefs. Green dots, which always lie on the red model curve show the model predictions of the data (blue triangles). The light-blue area shows the belief region where the ordered logistic model predicts the listen action. In the red and green areas the ordered logistic model predicts Open Left and Open Right action respectively. The absence of these areas in this model suggests that the model without the social information fails to predict the observed open left/right actions. (B) This model behavior shows the prediction made with the social information (Eq 2). This model predicts most of the OL actions (red area) and OR actions (green area) correctly demonstrating the importance of the social information (CR/CL) for correctly predicting the observed data.

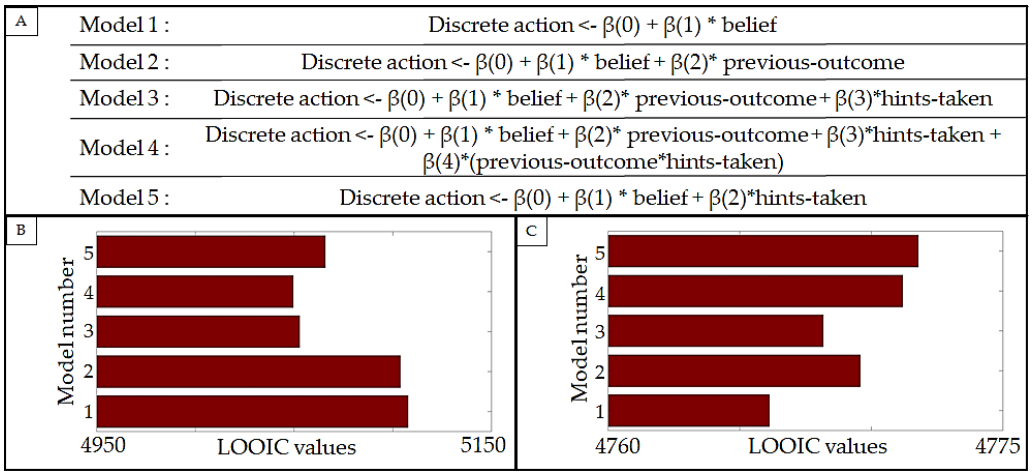


Figure 4: (A) Different models compared of the multi-player version of the task. All the models in (B) without the social information perform worse compared with the LOOIC values of the models with the social information added in (C) (see different scales in (B) and (C)). The simplest model with just the belief update (model number 1) in (C) performed better when compared to extensions of number of hints taken, previous outcome and an interaction of them.

Reinforcement Learning in the Acquisition of Unintuitive Motor Control Patterns

Sarah A. Wilterson
Department of Psychology
Princeton Neuroscience Institute
Princeton University
shutter@prineton.edu

Samuel D. McDougle
Department of Psychology
University of California – Berkeley
mcdougle@berkeley.edu

Jordan A. Taylor
Department of Psychology
Princeton Neuroscience Institute
Princeton University
jordanat@princeton.edu

Abstract

Development of a mapping between goals and actions (i.e. learning a motor controller) has been shown to rely on processes previously associated with decision-making and reinforcement learning. However, the link between reinforcement learning and motor skill has not been thoroughly explored. Here, we sought to probe this potential link by biasing learning in a motor task toward model-free and model-based processes to determine if doing so would shape the eventual motor controller. Subjects were trained to navigate a cursor across a grid using an arbitrary and unintuitive mapping. We hypothesized that knowledge of the correct sequence would tip the balance between the reinforcement learning processes, which would be revealed in a transfer test. When subjects were tasked with navigating to a novel set of start-end locations, those who learned the sequence without the benefit of explicit instruction far outperformed subjects given the correct key-press sequence. Consistent with learning arising from a model-free process, the group given additional information in training did not fully learn the mapping between their finger-presses and the resultant on-screen movement. In the next experiment, we call into question the ability to use this newly learned controller to plan future states; however, this may depend on the expertise with the novel mapping. In a follow-up experiment, the complexity of the newly trained mapping interacted with the amount of prior learning to determine how far into the future subjects could plan. Additionally, we found that reaction time increased as a function of the number of planned states, indicative of a computationally-demanding process typically associated with model-based planning. We conclude that flexibility of a new controller is at least partially determined by initial learning conditions, and that the ability to plan ahead requires extensive training.

Keywords: Motor Control, Skill Learning, Transfer of Learning

Acknowledgements

This work was supported by the National Institute of Neurological Disorders and Stroke (R01 NS-084948)

1. Introduction

Humans complete a wide variety of motor tasks on a daily basis. Some tasks are relatively simple and familiar, such as climbing a flight of stairs, while others require learning a complicated set of novel movements, like learning to play a new piano piece. The effective control of movement is itself a challenging task, and the challenge only increases as our goals become progressively more variable or complex. The job of specifying how to coordinate the muscles and limbs in order to achieve a goal is given to the motor controller - a computational module that takes as its input a desired goal or physical state and returns a plan for realizing that goal. The complexity of this computation has inspired debate surrounding the rules for establishing and modifying a feedback controller. Much investigation of motor controllers is predicated on system identification techniques borrowed from engineering. With these techniques, a system is characterized by recording its responses to systematic perturbations [1]. This research has been entrenched in showing when and how an established controller, particularly that responsible for reaching and grasping, adapts in the face of perturbations (e.g. [2, 3, 4]).

Unlike the adaptation paradigms, tasks which develop a new motor controller must require a novel mapping of actions onto desirable goal states. This is sometimes referred to as "skill learning," and has received relatively little attention from a control theory prospective [5, 6]. It has been proposed that skill learning begins with effortful, carefully evaluated movements and continues until action sequences are accomplished without the express need of an agent's attention [7]. Historically, this has been thought of as a progression, with early, effortful processing giving way to fast, automatic processes [5]. Although this idea has been around for over 50 years, we have no real mechanistic understanding for the progression of skill learning.

A parallel to this motor learning progression has been carefully studied in the field of reinforcement learning. The terminology here is different, but the description of this progression is familiar, with early, goal-directed behavior consolidating into habitual actions after practice [8, 9, 10]. Indeed, the model-based stage of learning is often characterized as slow and effortful, requiring a computationally intensive tree search of action choices [11], echoing the attention-demanding effort of early skill learning. On the other hand, once learning becomes model-free, it is no longer able to search the decision tree for the best solution. This results in characteristically rigid performance. Nonetheless, the response is quickly and automatically accessible without need for effortful processing or attention [11]. We recognize this as a close description of a learned skill, which can be completed rapidly without effortful control.

There is some experimental evidence suggesting that model-free and model-based learning also underlie learning of a motor controller. Particularly, imposing time pressure on subjects when they are learning an unintuitive keyboard-to-cursor mapping results in suppression of their ability to generalize the learning to new target locations [12]. However, these participants were able to generalize to new target locations when provided sufficient time to prepare. This suggests that normal use of a motor controller relies on a time-consuming computation, such as a model-based process. A model-based controller could afford generalization by rolling out the consequences of various choices using learned action-outcome associations. However, imposing a time-limit on the motor controller may have forced reliance on faster, but less flexible, model-free learning.

Over three experiments, we sought to determine parallels between established features of reinforcement learning and acquisition of a novel controller. We began by biasing learning toward either model-free or model-based processes and measuring the flexibility of the newly learned controller. In the second and third experiments, we investigate the ability of the motor system to use a newly learned controller to plan out future states.

2. Experiments and Results

We modified a task developed by Fermin and colleagues (2010), in which subjects navigate a cursor across a virtual grid using a keyboard [12]. The mapping between key-presses and movement of the cursor was arbitrary and unintuitive, which is thought to require learning *de novo*. In our variant of this task, subjects navigated a cursor across a grid by pressing keys on a keyboard. The mapping between key-presses and movement of the cursor was arbitrary and differed across subjects. Subjects practiced navigating between a single start-end pair (training phase) and were tested on multiple, novel start-end pairings (test phase). The exact process and number of these tests varied by experiment.

2.1 Experiment 1 – Instruction Biases Learning to Model-Based Processes

In Experiment 1, subjects trained on an arbitrary key-response mapping deterministically tied to three keyboard keys (e.g. 'D' = up, 'F' = right, 'J' = left). All twenty training trails featured the same start and end position, which was always a six key-press path and pseudorandomized across participants. To bias learning toward model-free or model-based processes, we manipulated the instruction that subjects were given about how to solve the task. In the Instruction group, subjects were given the exact sequence of key-presses which would lead them to the target (e.g. "Press J-J-J-D-F-F"). In contrast, subjects in the No Instruction group were not provided with specific instructions for executing the path, forcing them to explore the mapping between key-presses and cursor movement during training. We hypothesized that knowledge of the correct sequence would tip the balance between different learning processes, which would be revealed with a transfer test where subjects navigated to a novel set of start-end locations.

On the transfer test, subjects were given seven novel start/end location pairs to navigate. One trial with the trained start/end location was mixed randomly into the transfer test.

We found that providing full knowledge of the required key-press sequence greatly speeded initial learning (Figure 1A). However, on the transfer test, subjects who learned the sequence by trial and error far (Instruction Group) outperformed subjects given prior knowledge (No Instruction Group, Figure 1B). We take this as evidence that the No Instruction group produced a more flexible, model-based controller. Several control experiments confirmed that differences between the groups were not attributable to variability of experience in training, working memory, or explicit knowledge of the controller.

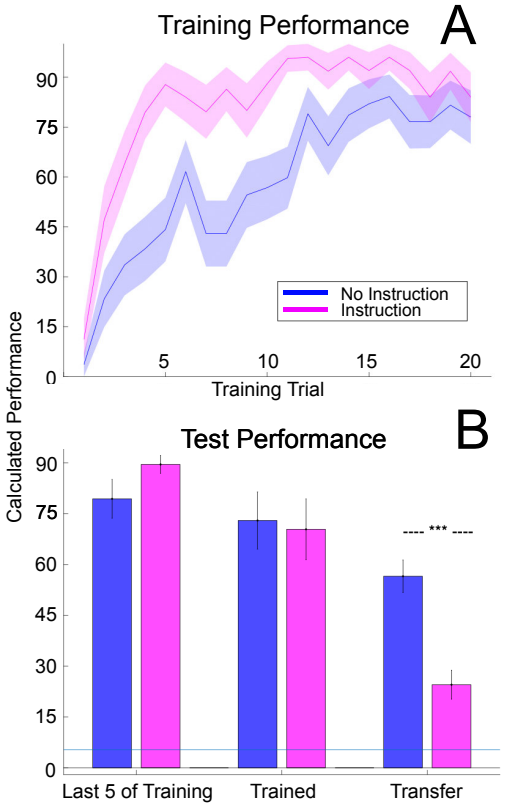


Figure 1. Average points out of a hundred for fifty subjects; points were lost for not hitting the target (-100) and taking additional steps past what was necessary (-5/per). Shaded regions/error bars represent standard error. Horizontal line represents chance. (A) Time course of learning during training. B) Last five training trials, the start/end location that was trained, and the novel, transfer, start/end locations.

2.2 Experiment 2 – The Newly Learned Controller Requires State-Space Feedback

Next, we sought to determine how well a newly learned controller could be used to plan out a set of future states (i.e., a route between novel start-end locations) by removing continuous feedback of the cursor. The training phase was identical to that of Experiment 1, again with two groups: Instruction and No Instruction. The only difference from the test phase of Experiment 1 was the absence of continuous positional feedback. Subjects were able to see their position at the beginning of each trial, but once they began moving their curser disappeared and they had to complete the trail "in the dark". Positive feedback was given if the subject hit the target space.

With feedback removed in the test phase, performance on transfer trials dramatically declined to at or just-above chance (Figure 3). This finding calls into question the ability to use a novel, feedback controller to plan future states. This is true of both the group biased to be model-free and the group biased to be model-based. We hypothesized that subjects are unable to plan out future states using the newly learned controller because each blind movement greatly increases the number of possibilities for one's current location. Even a subject with 80% confidence in any given movement would quickly face an unmanageable level of uncertainty five movements into the task.

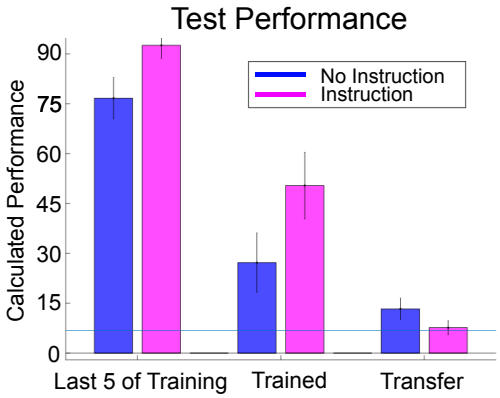


Figure 2. Average points out of a hundred for fifty subjects on the last five training trials, the start/end location that was trained, and the novel, transfer, start/end locations.

2.3 Experiment 3 – Extended Practice allows for Greater Flexibility without Feedback

In Experiment 3, we tested whether the ability to plan without state-space feedback would depend on the complexity of the mapping and the degree of expertise. Unlike Experiments 1 and 2, the test trials in Experiment 3 were interleaved through-out training. Subjects would complete five training trails, again with the same start/end pair, and then be given a single, no-feedback transfer trial. Each transfer trial consisted of a set of start-end locations pseudorandomly chosen to be 1, 2, 3, or 4 key-presses apart. This was repeated for 140 training trials, with subjects receiving one transfer trial of each length at least once every 20 training trials. Each subject completed the full experiment once with a 3-key mapping and once with a more difficult 6-key mapping.

The complexity of the mapping and degree of learning interacted to determine how far in advance subjects could plan (Figure 4A). What's more, we found that subjects' reaction times increased with the number of planned future states (Figure 4B). We conducted an additional experiment to control for the presence of test trials during training. This follow-up experiment provided the same results as shown above. These results suggest that with extended practice a feedback controller can be co-opted to plan ahead, but doing so is computationally demanding.

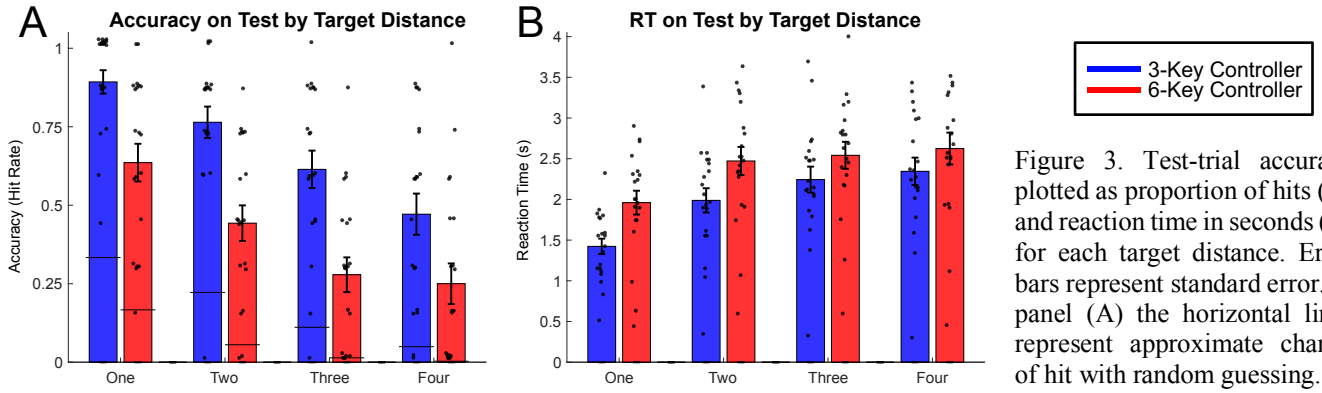


Figure 3. Test-trial accuracy plotted as proportion of hits (A) and reaction time in seconds (B) for each target distance. Error bars represent standard error. In panel (A) the horizontal lines represent approximate chance of hit with random guessing.

3 Discussion

The goal of this study was to determine the extent of the analog between reinforcement learning in decision making and motor skill acquisition. Across three experiments, we found that participants can leverage both model-free and model-based processes to improve performance in a novel motor task. In the first experiment, we reveal that prior knowledge when learning a new motor controller determines behavioral flexibility when conditions change. Use of the new controller to navigate novel start/end locations was impaired when explicit knowledge of the correct training sequence was provided. We interpret this inability to transfer knowledge as the consequence of biasing early learning toward a model-free process, which resulted in weakened acquisition of the underlying motor controller. In Experiment Two we show that the model-based learning that allowed transfer in Experiment One is insufficient for a complete tree search of future movements. However, in Experiment Three we find that the ability to roll-out a complete motor plan is possible, and dependent on sufficient experience with the motor controller. Completing the full tree search necessary to formulate this motor plan is computationally taxing, as evidenced by increased reaction times when the number of planned steps increases. Planning with a controller requires extensive training, which reflects the commonly arduous experience of mastering a new motor skill. Our results, and a preliminary modeling analysis, are consistent with the idea that a model-based process underlies effective learning of a motor controller.

To be entirely consistent with reinforcement learning theory, model-based processes should eventually give-way to model-free processes in development of a motor controller. We anticipate the next evolution of the controller to include some caching of responses, such that there isn't a reaction time cost for its use. This would represent the highest level of skill in motor control.

References

[1] Thoroughman, K. A., & Shadmehr, R. (2000). Learning of action through adaptive combination of motor primitives. *Nature*, 407(6805), 742.

[2] Pine, Z. M., Krakauer, J. W., Gordon, J., & Claude, G. C. (1996). Learning of scaling factors and reference axes for reaching. *NeuroReport*, 7, 2357-2361.

[3] Fine, M. S., & Thoroughman, K. A. (2006). Motor adaptation to single force pulses: sensitive to direction but insensitive to within-movement pulse placement and magnitude. *Journal of neurophysiology*, *96*(2), 710-720.

[4] Fine, M. S., & Thoroughman, K. A. (2007). Trial-by-trial transformation of error into sensorimotor adaptation changes with environmental dynamics. *Journal of Neurophysiology*, 98(3), 1392-1404.

[5] Fitts, P. M., & Posner, M. I. (1967). Human performance. Wadsworth Publishing Co. Belmont, CA.

[6] Haith, A. M., & Krakauer, J. W. (2013). Model-based and model-free mechanisms of human motor learning. In Progress in motor control (pp. 1-21). Springer, New York, NY.

[7] Crossman, E. R. F. W. (1959). A theory of the acquisition of speed-skill. Ergonomics, 2(2), 153-166.

[8] Adams, C. D. (1982). Variations in the sensitivity of instrumental responding to reinforcer devaluation. *The Quarterly Journal of Experimental Psychology Section B*, 34(2b), 77-98.

[9] Dickinson, A. (1985). Actions and habits: the development of behavioural autonomy. *Phil. Trans. R. Soc. Lond. B*, 308(1135), 67-78.

[10] Killcross, S., & Coutureau, E. (2003). Coordination of actions and habits in the medial prefrontal cortex of rats. *Cerebral cortex*, 13(4), 400-408.

[11] Daw, N. D., Niv, Y., & Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12), 1704.

[12] Fermin, A., Yoshida, T., Ito, M., Yoshimoto, J., & Doya, K. (2010). Evidence for model-based action planning in a sequential finger movement task. *Journal of motor behavior*, 42(6), 371-379.

Rate-Distortion Theory and Computationally Rational Reinforcement Learning

Rachel A. Lerch & Chris R. Sims
Department of Cognitive Science
Rensselaer Polytechnic University
Troy, NY 12180
lerchr2@rpi.edu — simsc3@rpi.edu

Abstract

We examine reinforcement learning (RL) in settings where there are information-theoretic constraints placed on the learner's ability to encode and represent a behavioral policy. This situation corresponds to a challenge faced by both biological and artificial intelligent systems that must seek to act in a near-optimal fashion while facing constraints on the ability to process information. We show that the problem of optimizing expected utility within capacity-limited learning agents maps naturally to the mathematical field of rate-distortion (RD) theory. RD theory is the branch of information theory that provides theoretical bounds on the performance of lossy compression. By applying the RD framework to the RL setting, we develop a new online RL algorithm, Capacity-Limited Actor-Critic (CL-AC), that optimizes a tradeoff between utility maximization and information processing costs. Using this algorithm in a discrete gridworld environment, we first demonstrate that agents with capacity-limited policy representations naturally avoid "policy overfitting" and exhibit superior transfer to modified environments, compared to policies learned by agents with unlimited information processing resources. Second, we introduce a capacity-limited policy gradient theorem that enables the extension of our approach to large-scale or continuous state spaces utilizing function approximation. We demonstrate our approach using the continuous Mountain Car task.

Keywords: Reinforcement Learning, Information Theory,
Rate-Distortion Theory, Computational Rationality

Acknowledgements

This research was supported by NSF grant DRL-1560829 and a grant from the RPI-IBM Artificial Intelligence Research Collaboration (AIRC). We would like to thank Matt Reimer, Tim Klinger, Miao Lui & Gerald Tesauro at IBM TJ Watson Research Center for their helpful comments and feedback on this work.

1 Introduction

We consider the problem of capacity-limited RL, in which learning agents lack the ability to store or represent behavioral policies with perfect fidelity. This work is motivated by the framework of computational rationality [1], an emerging paradigm for understanding biological and artificial intelligence as optimizing performance subject to constraints on information processing. Previous research has identified fundamental information processing limits on human reinforcement learning [2]. Building on this work, we examine constraints on the policy representation of the learner, formally defined in the information theoretic sense, and demonstrate the implications of such constraints on self-guided learning.

In the standard RL setting [3], the agent's goal is to learn an optimal policy, $\pi^*(a \mid s)$. In the present work, we consider this policy function as an *information channel*, that takes as input a current state, and produces an action to be followed (e.g., $\mathcal{S} \rightarrow \mathcal{A}$). To limit the amount of information stored about the policy or environment, we apply the framework of rate-distortion (RD) theory [4]. Within RD theory, an information channel is abstractly modeled as a conditional probability distribution $p(y \mid x)$, which describes the probability of an input signal x, drawn from a source distribution $p(x)$, producing an output signal y. Optimal performance for this channel is defined by a loss function, $\mathcal{L}(x, y)$ that quantifies the cost of a signal $x \in X$ being transmitted as the value $y \in Y$. The goal for the channel is to minimize the cost of communication error specified by the expected loss, $\mathrm{E}\left[\mathcal{L}(x, y)\right]$. Lastly, for a given source $p(x)$ and channel distribution $p(y \mid x)$, information theory provides a measure of the amount of information communicated (on average) by the channel in terms of its mutual information, $\mathrm{I}(X, Y)$.

Any physical communication channel is necessarily limited to transmitting information at a finite rate. Consequently, an optimally efficient communication channel is one that minimizes expected loss subject to this constraint:

$$\textbf{Goal: } \text{Minimize } \mathrm{E}\left[\mathcal{L}(x, y)\right] \text{ w.r.t. } p(y \mid x), \text{ subject to } \mathrm{I}(X, Y) \leq C. \tag{1}$$

In [5], Blahut developed an efficient iterative algorithm for solving this problem for channels with discrete input and output alphabets. The Blahut algorithm minimizes the combined performance objective $\mathrm{I}(X, Y) + \beta \mathrm{E}\left[\mathcal{L}(x, y)\right]$ with respect to $p(y \mid x)$, where the parameter $\beta > 0$ controls the tradeoff between the expected loss and information rate of the channel.

Rather than being concerned with the mapping of abstract communication signals x and y, in the RL setting we are interested in a channel that is concerned with the mapping from states to actions (policy mapping). In the present work, we limit our attention to exploring the impact of capacity limits on the policy mapping, although in principle the same approach could be extended to the value function as well. A critical component in the application of RD theory to RL is the specification of an appropriate loss function for a capacity limited policy channel. To that end, we introduce the *Bellman loss function*:

$$\mathcal{L}^\star(s, a) = \max_{a'} q^\star(s, a') - q^\star(s, a) \tag{2}$$

$$= v^\star(s) - \sum_{s', r} p(s', r \mid s, a)(r + \gamma v^\star(s')). \tag{3}$$

This quantity is, by definition, the loss in expected utility associated with starting in state s and following action a, relative to the best possible action for that state. Intuitively, the Bellman loss function says that when there is little difference between the long-term cost of actions in a given state, there is no need to precisely encode a policy that distinguishes between them. As we will demonstrate, minimizing the Bellman loss subject to an information constraint also motivates exploratory behavior, without requiring an added intrinsic curiosity-based reward signal.

2 Properties of Capacity-Limited Policies

In this section, we demonstrate the implications of adopting a capacity limited policy in a simple 2D gridworld (Figure 1A). The goal in this task is to navigate from a starting state to a terminal goal position. The states ($\mathcal{S}$) are defined by the current location of the agent within a 12x12 maze. The available actions ($\mathcal{A}$) correspond to taking a step in one of four cardinal directions (north, south, east, west). The agent incurs a 1-point penalty per step, and a 10-point penalty for colliding with walls. Hence the objective is finding a least-cost path to the target.

The Bellman loss function defined in the previous section requires knowledge of the optimal value function. We start by assuming that $V^\star(s)$ is known (solved via dynamic programming). The goal of the demonstration is to explore the resulting changes in behavior as capacity limits are varied on the policy function $\pi(a \mid s)$.

As an intuitive example, one can consider two kinds of behavioral policies for a gridworld environment. In one case, there are no information constraints, and the agent simply remembers the optimal action associated with each state. As the size of the state or action space grows, or for physical agents with limited computational resources, such a policy may be infeasible to store with perfect fidelity. In the case of information constraints, a 'compressed' policy might consist of a general plan (with high probability, move north or east) while storing more detailed instructions for key states where there are high costs for error.

To explore this idea, we apply the Blahut algorithm to obtain compressed policy channels at various levels of channel capacity. Figure 1B illustrates the fundamental tradeoff between channel capacity and performance. In this gridworld

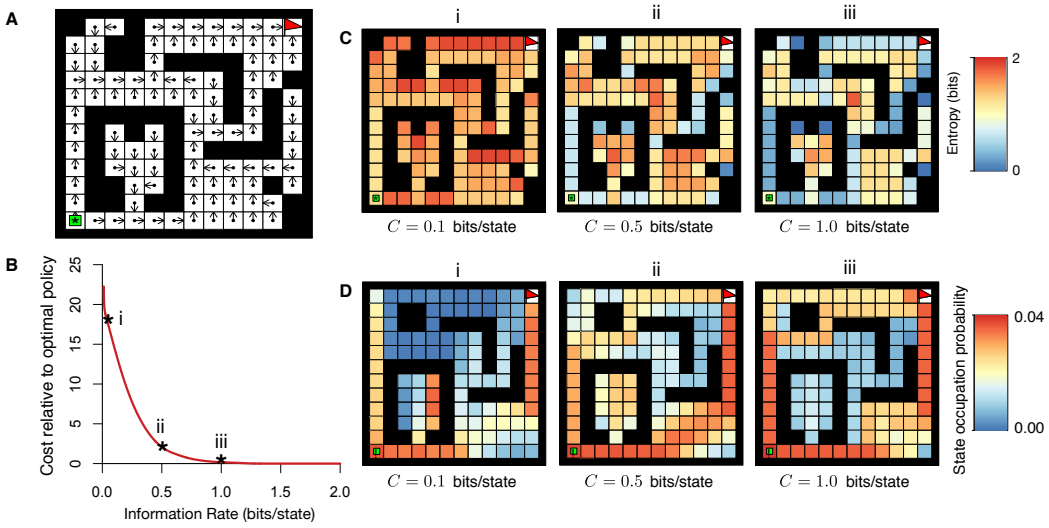


Figure 1: **A**) Optimal deterministic policy for each state. **B**) Each point along the curve represents an optimal policy that achieves the maximal expected value at a given rate of information. **C**) 3 policies illustrated at different points along the information rate distortion tradeoff curve (i–iii). Colors in the plots illustrate the entropy of the policy in each state of the maze **D**) Average probability of occupying each state for learned policies (online) using 3 different information rate constraints for a similar maze (averaged over 1,000 episodes).

environment, specifying an optimal deterministic policy requires 2 bits per state (the maximum entropy with 4 possible actions). The results illustrate that behavioral policies can be substantially compressed (by over half) without incurring significant cost to the agent.

Figure 1C illustrates three different capacity-limited policies ($C = 0.1, 0.5, 1.0$ bits per state, on average), indicated by the labeled plot markers in the bottom left plot. The colors for each state indicate the entropy of the policy in that state. With very limited capacity the agent's policy is near-random. With increasing capacity, the agent focuses its limited computational resources on representing important features of the environment with high fidelity (e.g. learning how to navigate key corridors), while using a stochastic policy in open areas of the maze. Notably, Figure 1C also illustrates that a form of exploration/exploitation tradeoff also emerges automatically from capacity-limited policies: behavior is naturally more stochastic (exploratory) in states where the costs of error are unknown, or known to be small. Lastly, Figure 1D illustrates the probability of occupying each state according to each of the three policies. At very low information rates, a policy encodes little more than "move up and right", and consequently becomes trapped in corners of the maze with high probability (Figure 1D(i)).

3 On-line Learning via Capacity-Limited Actor-Critic (CL-AC)

In this section, we consider the problem of simultaneously learning a value function online, and from it gradually improving a capacity-limited policy. We develop our algorithm using the Actor-Critic (AC) framework in RL. Because of the recursive dynamics between an agent's current policy, future exploration, and updated policy, it is not *a priori* obvious that a capacity-limited agent will be able to discover or learn an effective policy.

Computing the Bellman loss function requires knowledge of the optimal value function for the task, $v^\star(s)$. The approach we adopt is simply to substitute an estimate of the value function, $v(s)$, which is learned in an on-line manner using standard temporal difference (TD) learning. The estimated loss function $\mathcal{L}(s, a)$ is updated via the same temporal difference error. The required elements for learning are a starting state and action, (s, a), the observed sample reward r and resulting state s', along with the current estimate of the value function. This yields:

$$\mathcal{L}(s, a) \leftarrow \mathcal{L}(s, a) + \eta \left[v(s) - (r + \gamma v(s')) - \mathcal{L}(s, a) \right]. \tag{4}$$

Note that the first two terms inside the square brackets are online samples of the Bellman loss function (Eq. 3), substituting the estimated value function in place of the optimal value function. In the current work a common learning rate parameter is adopted for both the value and loss function. With an estimated loss function (Eq. 4), it is possible to obtain an optimal capacity-limited channel for that loss function using the Blahut algorithm [5].

The CL-AC algorithm was tested on the gridworld environment introduced previously, with varying values of the parameter β. For each, performance was averaged across 2,500 randomly generated maze environments. The learning rate was fixed at $\eta = 0.1$, and no temporal discounting was assumed. The results are shown in Figure 2, left panel. As

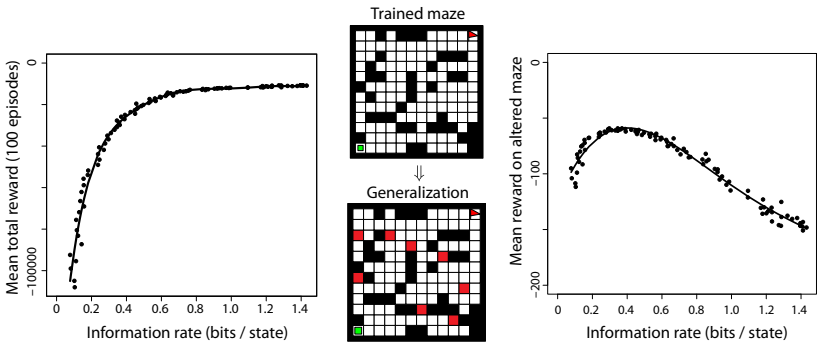


Figure 2: **Left**: Average accumulated reward for CL-AC across 100 training episodes. **Middle**: Example training and generalization environments. Added walls are highlighted in red. **Right**: Average generalization performance for CL-AC, as a function of the trade-off parameter β.

before, we demonstrate that the representation of a behavioral policy can be extremely low-fidelity while still allowing near-optimal performance.

In machine learning, it is commonly understood that complex models run the risk of *overfitting*: good performance on a training set, but generalizing poorly to new environments. This raises the question of whether RL agents also suffer from overfitting in terms of their learned policies, and whether capacity-limited RL could alleviate this problem. To test this idea, we trained RL agents in randomly constructed gridworlds for 100 episodes. At the end of training, we modified the maze by adding 8 additional walls placed in random locations, subject to the constraint that a viable path from the start state to terminal state existed (2, middle). We compared the performance of the CL-AC algorithm using 100 different values of the capacity tradeoff parameter β, averaged across 2,500 random maze environments. Generalization performance was evaluated in terms of the expected value of the learned policy, as applied to the altered gridworld maze.

Intuitively, one might think that capacity-limited agents should always be outperformed by higher capacity agents that are able able to represent their policies with greater fidelity. Our results demonstrate that this is not the case. Figure 2, right panel, illustrates that generalization performance is highest at intermediate levels of channel capacity (0.5 bits per state). An intuitive account for this result lies in the concept of policy overfitting. Capacity limits force learning agents to concentrate representational resources on critical states, at the expense of increasing stochasticity in states where the costs for error are less critical. Consequently, when particular paths are blocked, the capacity-limited agents retain exploratory policies that are likely to be viable alternative solutions. These results reiterate the benefits imparted by capacity-limited policies, as they naturally impart regularization in learning systems towards policies that are robust, and more effectively generalize past experiences to new environments.

4 Extension to Continuous State Spaces

In continuous or complex environments, modern RL approaches require some form of function approximation (such as a neural network) to represent value functions and policies [3]. In order to extend CL-AC to this setting, we have developed a novel stochastic gradient descent (SGD) version of the Blahut algorithm. Following the derivation of the standard policy gradient theorem [3], we introduce a parameterized policy, $\pi_\theta(a \mid s)$, and define a performance objective that our policy should seek to optimize with respect to θ. The objective function reflects the fundamental RD tradeoff between information rate and expected loss associated with following the policy: $J(\theta) = (1-\beta)\mathrm{I}(s,a) + \beta \mathrm{E}\left[\mathcal{L}(s,a)\right]$.

Note that we have re-parameterized β, such that when $\beta = 1$ the objective is equivalent to unconstrained utility maximization. Whereas Blahut developed an efficient coordinate descent algorithm for minimizing this objective in the discrete setting, we instead adjust the continuous parameters of the policy θ via gradient descent: $\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$. As computing the mutual information requires summing or integrating over the full state space, we instead perform stochastic gradient descent by sampling states according to the on-policy distribution, and computing the local gradient at each state. Mathematical analysis (manuscript in preparation) shows that our approach converges in expectation to a minimum of the objective, corresponding to an optimal, but capacity-limited policy.

In the present paper, we offer an empirical demonstration of the approach using the OpenAI Gym "Mountain Car" environment[1]. In this environment, an underpowered car must climb a hill by rocking back and forth in order to build up sufficient momentum (Fig. 3A). The state space corresponds to the continuous position and velocity of the car. The agent has three available actions: move left, neutral, or move right. On each time step the agent incurs a cost of -1 until the car reaches the goal at the top of the hill on the right.

[1] https://gym.openai.com/envs/MountainCar-v0/

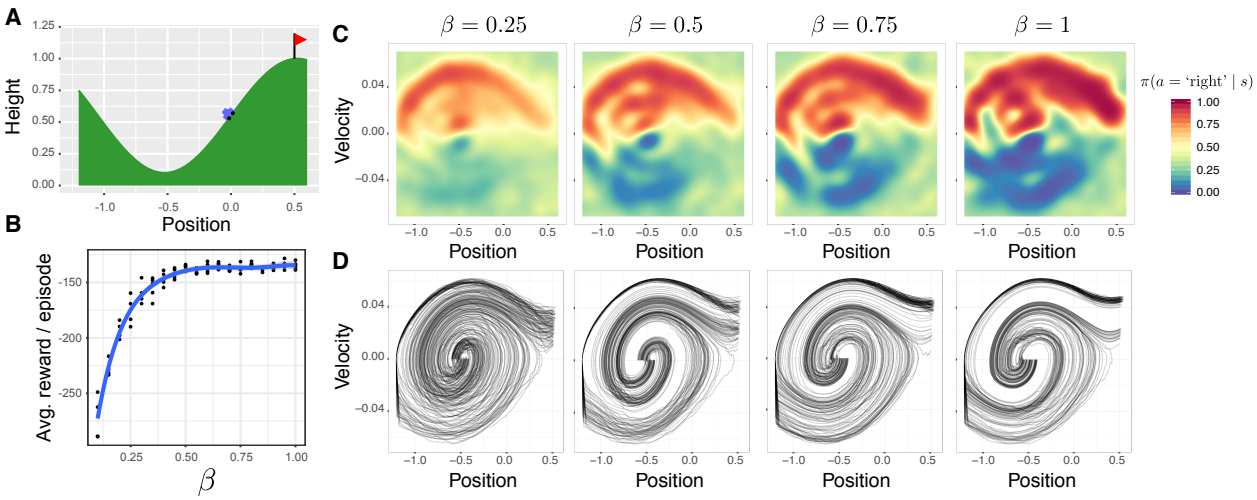


Figure 3: **A**: The Mountain Car task environment. **B**: Policy performance (expected reward per episode) after 500 training episodes with varying levels of β. **C**: Learned policies (shown as the probability of selecting the action 'right' in each state) at 4 levels of the parameter β. **D**: 100 sample episode trajectories generated from the policies illustrated in C.

Our model for the Mountain Car task used a radial basis function (RBF) network to encode the state, with a grid of 50×50 Gaussian units uniformly tiling the state space. A state-action value function $q(s, a)$ was represented using a linear combination of these features, with feature weights learned using the standard SARSA learning rule [3]. The policy $\pi_\theta(a \mid s)$ was implemented as a softmax layer with three output units corresponding to the three actions, receiving input from the same underlying RBF feature representation. Paralleling the standard Actor-Critic framework, the policy parameters were updated via gradient descent on each time step to minimize the (capacity-limited) performance objective.

Figure 3B illustrates the expected reward associated with varying levels of the parameter β after completing 500 training episodes. As with the gridworld environment, the results show that the fidelity of an agent's policy can be substantially reduced without incurring excessive cost. We believe this may reflect a common property of natural environments, where there are often large regions of the state space where optimal policies are either irrelevant or no better than random choice. Computationally rational agents can capitalize on and exploit these natural task statistics, analogous to the idea of efficient coding in perception. Figure 3C illustrates four different policies, for agents with $\beta = \{0.25, 0.5, 0.75, 1.0\}$, the latter corresponding to an unlimited capacity agent. Panel D, below, illustrates 100 trajectories through the state space generated by each of these policies. Much as in the gridworld environment, higher information rates yield increasingly deterministic policies, but without an accompanying increase in utility to the agent. In future work we plan to explore the generalization ability of capacity-limited policies in the mountain car environment, as well as in larger scale environments.

5 Summary

This paper describes the application of RD theory to the learning of limited yet *efficient* behavioral policies. We show that such policies enable superior generalization, and naturally impart principled exploratory behavior. RD theory enforces a budget on deterministic (exploitative) policies, and channels exploration intelligently. We further demonstrate the extension of our approach to continuous state spaces. In aggregate, this body of work demonstrates the value of a principled framework for the development and specification of computationally rational learning agents.

References

[1] S. J. Gershman, E. J. Horvitz, and J. B. Tenenbaum, "Computational rationality: A converging paradigm for intelligence in brains, minds, and machines," *Science*, vol. 349, no. 6245, pp. 273–278, 2015.

[2] A. G. Collins and M. J. Frank, "How much of reinforcement learning is working memory, not reinforcement learning? A behavioral, computational, and neurogenetic analysis," *European J. of Neuroscience*, vol. 35, no. 7, pp. 1024–1035, 2012.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2nd ed., 2018.

[4] T. Berger, *Rate distortion theory: A mathematical basis for data compression*. Prentice-Hall, 1971.

[5] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

Searching for Markovian Subproblems to Address Partially Observable Reinforcement Learning

Rodrigo Toro Icarte
Department of Computer Science
University of Toronto & Vector Institute
Toronto, Ontario, Canada
rntoro@cs.toronto.edu

Ethan Waldie
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
ethan.waldie@mail.utoronto.ca

Toryn Q. Klassen
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
toryn@cs.toronto.edu

Richard Valenzano
Element AI
Toronto, Ontario, Canada
rick.valenzano@elementai.com

Margarita P. Castro
Department of Mechanical and Industrial Engineering
University of Toronto
Toronto, Ontario, Canada
mpcastro@mie.utoronto.ca

Sheila A. McIlraith
Department of Computer Science
University of Toronto & Vector Institute
Toronto, Ontario, Canada
sheila@cs.toronto.edu

Abstract

In partially observable environments, an agent's policy should often be a function of the history of its interaction with the environment. This contradicts the Markovian assumption that underlies most reinforcement learning (RL) approaches. Recent efforts to address this issue have focused on training Recurrent Neural Networks using policy gradient methods. In this work, we propose an alternative – and possibly complementary – approach. We exploit the fact that in many cases a partially observable problem can be decomposed into a small set of individually Markovian subproblems that collectively preserve the optimal policy. Given such a decomposition, any RL method can be used to learn policies for the subproblems. We pose the task of learning the decomposition as a discrete optimization problem that learns a form of Finite State Machine from traces. In doing so, our method learns a high-level representation of a partially observable problem that summarizes the history of the agent's interaction with the environment, and then uses that representation to quickly learn a policy from low-level observations to actions. Our approach is shown to significantly outperform standard Deep RL approaches, including A3C, PPO, and ACER, on three partially observable grid domains.

Keywords: Partial Observability
Reinforcement Learning
Automata Learning
Reward Machines

Acknowledgements

We gratefully acknowledge funding from CONICYT (Becas Chile), the Natural Sciences and Engineering Research Council of Canada (NSERC), and Microsoft Research.

1 Introduction

Partially observable environments remain very challenging for RL agents because they break the Markovian assumption with respect to the agent's observations. As a result, agents in these environments require some form of memory to summarize past observations. Recent approaches either encode the observation history using recurrent neural networks [5, 10, 7] or use memory-augmented neural networks to provide the agent access to external memory [6]. We propose an alternative approach that searches for a decomposition of the task into a small set of individually Markovian subtasks.

For example, consider the 2-keys domain shown in Figure 1c. The agent (purple triangle) receives a reward of +1 when it reaches the coffee machine, which is always in the yellow room. To do so, it must open the two doors (shown in brown). Each door requires a different key to open it, and the agent can only carry one key at the time. At the beginning of each episode, the two keys are randomly located in either the blue room, the red room, or split between them. Since the agent can only see what is in the current room, this problem is partially observable.

This problem is quite difficult for current RL approaches: A2C, ACER, and PPO performed poorly on this task even after 5 million training steps (Section 4). However, it is decomposable into a small set of Markovian subproblems. The first involves searching for the keys. Notice that if the agent finds only one key in the red room, then (if it has learned enough about the domain) it can deduce that the second key is in the blue room. The next subproblem is to pick up a key. This is followed by a subproblem involving opening a door and retrieving the other key. Crucially, this last subproblem is Markovian because the agent already knows which room the key is in based on which key they previously picked up.

Main contributions: We propose a discrete optimization-based approach that finds a high-level decomposition of a partially observable RL problem. This decomposition splits the problem into a set of Markovian subproblems and takes the form of a *reward machine (RM)* [8]. We also extend an existing method for exploiting RMs to the partially observable case, so that we can use a found RM to quickly learn a policy from low-level observations to actions. Finally, we show that our approach significantly outperforms several well-known policy gradient methods on three challenging grid domains.

Related work includes some early attempts to tackle partially observability in RL based on automata learning, e.g. [3, 4]. Both works rely on learning finite state machines at a low-level (over the environment observations). In contrast, our approach relies on learning a decomposition of the problem at the abstract level given by a labelling function. This allows our approach to also work over problems with continuous (or very large) observation spaces.

2 Preliminaries

A *Markov Decision Process (MDP)* is a tuple $\mathcal{M} = \langle S, A, r, p, \gamma \rangle$, where S is a finite set of *states*, A is a finite set of *actions*, $r : S \times A \to \mathbb{R}$ is the *reward function*, $p(s, a, s')$ is the *transition probability distribution*, and γ is the *discount factor*. The objective of $\mathcal{M}$ is to find a policy $\pi^* : S \to \Pr(A)$ that maximizes the expected discounted reward for every state $s \in S$. When r or p are unknown but can be sampled, an optimal policy can be found using RL approaches like *q-learning*. This *off-policy* method uses sampled experience of the form (s, a, s', r) to update $\tilde{q}(s, a)$, an estimate of the optimal *q-function*.

A *Partially Observable Markov Decision Process (POMDP)* is a tuple $\mathcal{P}_\mathcal{O} = \langle S, O, A, r, p, \omega, \gamma \rangle$, where S, A, r, p, and γ are defined as in an MDP, O is a finite set of *observations*, and $\omega(s, o)$ is the *observation probability distribution*. At every time step t, the agent is in exactly one state $s_t \in S$, executes an action $a_t \in A$, receives an immediate reward $r_{t+1} = r(s_t, a_t)$, and moves to the next state s_{t+1} according to $p(s_t, a_t, s_{t+1})$. However, the agent does not observe s_{t+1}, and only receives an observation $o_{t+1} \in O$ via ω, where $\omega(s_{t+1}, o_{t+1})$ is the probability of observing o_{t+1} from state s_{t+1} [1]. As such, many RL methods cannot be immediately applied to POMDPs because the transition probabilities and reward function are not necessarily Markovian w.r.t. O.

3 Learning to Decompose Partially Observable Problems

Our approach to RL in a partially observable environment has two stages. In the first, the agent solves an optimization problem over a set of traces to find a "good" *reward machine (RM)*-based [8] decomposition of the environment. In particular, we look for an RM $\mathcal{R}$ that can be used to make accurate one-step Markovian predictions over the traces in the training set. In the second stage, the agent uses any standard RL algorithm to learn a policy directly from low-level observations to actions for each subtask identified in $\mathcal{R}$. If at some point $\mathcal{R}$ is found to make incorrect predictions, additional traces are added to the training set and a new RM is learned. This process continues for as long as is desired.

Reward Machines under Partial Observability

Let us begin by defining RMs and identifying how a given RM can be used by an RL agent in a partially observable environment. RMs are finite state machines that give reward on every transition, and were recently proposed as a way to expose the structure of a reward function to an RL agent [8]. In the case of partial observability, RMs are defined over a set of propositional symbols $\mathcal{P}$ that correspond to a set of high-level features the agent can detect using a *labelling function*

$L : O_\emptyset \times A_\emptyset \times O \to 2^\mathcal{P}$ where $X_\emptyset = X \cup \{\emptyset\}$. L assigns truth values to symbols in $\mathcal{P}$ given an environment experience $e = (o, a, o')$ where o' is the next observation after executing action a from observation o. We use $L(\emptyset, \emptyset, o)$ to assign truth values to the initial observation. We call a truth value assignment over $\mathcal{P}$ an *abstract observation* since it provides a high-level view of the low-level observations via L. We now formally define an RM as follows:

Definition 3.1 (reward machine)**.** Given a set of propositional symbols $\mathcal{P}$, a set of (environment) observations O, and a set of actions A, a Reward Machine is a tuple $\mathcal{R}_{\mathcal{P}OA} = \langle U, u_0, \delta_u, \delta_r \rangle$ where U is a finite set of states, $u_0 \in U$ is an initial state, δ_u is the state-transition function, $\delta_u : U \times 2^\mathcal{P} \to U$, and δ_r is the reward-transition function, $\delta_r : U \times 2^\mathcal{P} \to \mathbb{R}$.

RMs decompose problems into high-level states U and define transitions using conditions defined by δ_u. These conditions are over a set of binary properties $\mathcal{P}$ that the agent can detect using L. For example, consider the RM for the 2-keys domain shown in Figure 1d. We assume that the agent can use L to detect the room color (🟥, ⬜, 🟦, and 🟨), the objects in the current room (🔑, ☕, and 🔒, where 🔒 represents a locked door), and whether it is carrying a key (▲). Each of these symbols is in $\mathcal{P}$. In the figure, we use "(🟥, ▲)" to denote that 🟥 and ▲ are true in the current state, and all other propositions (e.g. ⬜) are false. We also use "(🟥, ▲); (🟦, ▲)" to say that the transition is taken if either of these sets of propositions is satisfied. Finally, we note the figure only shows propositions sets that induce RM state changes. For all other sets, the RM simply remains in the same state it was in the last step.

The agent starts at the initial RM state u_0 and stays there until it observes the red room with no keys (🟥), one key (🟥,🔑) or two keys (🟥,🔑,🔑), or similarly for the blue room. Each of these conditions is associated with a unique arrow indicating the state to which the RM transitions. If the agent enters the blue room and there is one key (🟦,🔑), then the RM state changes from u_0 to u_1. The transitions in the RM are also associated with a reward via δ_r.

When learning policies given an RM, one simple approach is to learn a policy $\pi(o, x)$ that considers the current observation $o \in O$ and the current RM state $x \in U$. While a partially observable problem might be non-Markovian over O, it can be Markovian over $O \times U$ for some RM $\mathcal{R}_{\mathcal{P}OA}$. We call such an RM a *perfect RM*. For example, Figure 1d shows a perfect RM for the 2-keys domain given a labelling function that detects events 🟥, 🟦, 🔑, and ▲. It is perfect because it can correctly keep track of the locations of the keys once this is determined, which is all that the agent needs to remember in order to decompose the problem in a Markovian way. Formally, we define a perfect RM for POMDP $\mathcal{P}_\mathcal{O}$ as follows:

Definition 3.2. An RM $\mathcal{R}_{\mathcal{P}OA} = \langle U, u_0, \delta_u, \delta_r \rangle$ is considered *perfect* for a POMDP $\mathcal{P}_\mathcal{O} = \langle S, O, A, r, p, \omega, \gamma \rangle$ with respect to a labelling function L if and only if for every trace $o_0, a_0, \ldots, o_t, a_t$ generated by any policy over $\mathcal{P}_\mathcal{O}$, the following holds: $\Pr(o_{t+1}, r_{t+1} | o_0, a_0, \ldots, o_t, a_t) = \Pr(o_{t+1}, r_{t+1} | o_t, x_t, a_t)$ where $x_0 = u_0$ and $x_t = \delta_u(x_{t-1}, L(o_{t-1}, a_{t-1}, o_t))$.

Interestingly, we can formally show that if the set of belief states [1] for the POMDP $\mathcal{P}_\mathcal{O}$ is finite, then there exists a perfect RM for $\mathcal{P}_\mathcal{O}$. In addition, we can show that the optimal policies for perfect RMs are also optimal for $\mathcal{P}_\mathcal{O}$.

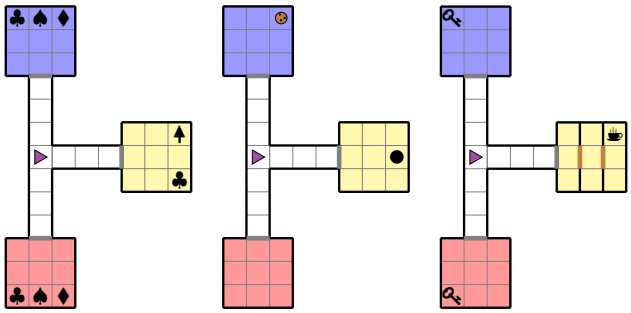
From Traces to Reward Machines

We now consider the problem of learning a perfect RM from traces, assuming one exists w.r.t. the given labelling function L. Since a perfect RM transforms the original problem into a Markovian problem over $O \times U$, we prefer RMs that accurately predict the next observation o' and the immediate reward r from the current observation o, the RM state x, and the action a. Instead of trying to predict the observations themselves, we propose a low-cost alternative which focuses on a necessary condition for a perfect RM: the RM must correctly predict what is *possible* and *impossible* at the abstract level given by L. It is impossible, for instance, to be at u_3 in the RM from Figure 1d and observe (🟥,🔑), because the RM is at u_3 iff the agent saw that the red room was empty or that both keys were in the blue room.

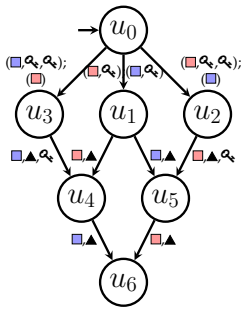
This idea is formalized in our optimization model (Figure 1e). Let $\mathcal{T} = \{\mathcal{T}_0, \ldots, \mathcal{T}_n\}$ be a set of traces, where each trace $\mathcal{T}_i$ is a sequence of observations, actions, and rewards: $\mathcal{T}_i = \{o_{i,0}, a_{i,0}, r_{i,0}, \ldots, o_{i,t_i}, a_{i,t_i}, r_{i,t_i}\}$. We now look for an RM $\langle U, u_0, \delta_u, \delta_r \rangle$ that can be used to predict $L(e_{i,t+1})$ from $L(e_{i,t})$ and the current RM state $x_{i,t}$, where $e_{i,t+1}$ is the experience $(o_{i,t}, a_{i,t}, o_{i,t+1})$ and $e_{i,0}$ is $(\emptyset, \emptyset, o_{i,0})$ by definition. The model parameters are the set of traces $\mathcal{T}$, the set of propositional symbols $\mathcal{P}$, the labelling function L, and a maximum number of states in the RM u_{max}. The model also uses the sets $I = \{0 \ldots n\}$ and $T_i = \{0 \ldots t_i - 1\}$, where I contains the index of the traces and T_i their time steps. The model has two auxiliary variables $x_{i,t}$ and $N_{u,l}$. Variable $x_{i,t} \in U$ represents the state of the RM after observing trace $\mathcal{T}_i$ up to time t. Variable $N_{u,l} \subseteq 2^\mathcal{P}$ is the set of all the next abstract observations seen from the RM state u and the abstract observations l at some point in $\mathcal{T}$. In other words, $l' \in N_{u,l}$ iff $u = x_{i,t}$, $l = L(e_{i,t})$, and $l' = L(e_{i,t+1})$ for some trace $\mathcal{T}_i$ and time t.

Constraints (2) and (3) ensure that we find a well-formed RM, while constraints (4) to (6) ensure that the found RM satisfies the current set of traces. Constraint (7) and (8) ensure that the $N_{u,l}$ sets contain at least every $L(e_{i,t+1})$ that has been seen right after l and u in $\mathcal{P}$. The objective function (1) comes from maximizing the log-likelihood for predicting $L(e_{i,t+1})$ using a uniform distribution over all the possible options given by $N_{u,l}$. A key property of this formulation is that any perfect RM is optimal with respect to the objective function in equation (1) when the number of traces (and their lengths) tends to infinity, if the traces are collected by a policy π such that $\pi(a|o) > \epsilon$ for all $o \in O$ and $a \in A$.

For solving this optimization problem, we found the local search algorithm *Tabu search* [2] to be effective. This method starts from an arbitrary feasible solution. It then iteratively examines all feasible "neighbouring" solutions, and moves



(a) Symbol domain. (b) Cookie domain. (c) 2-keys domain.



(d) A perfect RM.

$$\underset{\langle U,u_0,\delta_u,\delta_r\rangle}{\text{minimize}} \sum_{i\in I}\sum_{t\in T_i} \log(|N_{x_{i,t},L(e_{i,t})}|) \quad (1)$$

$$\text{s.t. } \langle U,u_0,\delta_u,\delta_r\rangle \in \mathcal{R}_{POA} \quad (2)$$

$$|U| \le u_{\max} \quad (3)$$

$$x_{i,t} \in U \qquad \forall i \in I, t \in T_i \quad (4)$$

$$x_{i,0} = u_0 \qquad \forall i \in I \quad (5)$$

$$x_{i,t+1} = \delta_u(x_{i,t}, L(e_{i,t+1})) \; \forall i \in I, t \in T_i \quad (6)$$

$$N_{u,l} \subseteq 2^{\mathcal{P}} \qquad \forall u \in U, l \in 2^{\mathcal{P}} \quad (7)$$

$$L(e_{i,t+1}) \in N_{x_{i,t},L(e_{i,t})} \qquad \forall i \in I, t \in T_i \quad (8)$$

(e) Optimization model for learning RMs.

Figure 1: Our domains, a perfect RM for the keys domain, and our optimization model.

to the neighbour with the best evaluation according to the objective function. For us, neighbouring RMs are defined as those that differ by exactly one transition. When a time limit is hit, the best seen solution is returned. Tabu search also maintains a set of states, the *Tabu list*, and prunes them from the "neighbouring" solutions to avoid revisiting them.

Finding Policies For Learned Reward Machines

Once we have learned an RM, we can use any RL algorithm to learn a policy $\pi(o,u)$, by treating the combination of o and u as the current state. However, doing so does not exploit the problem structure that is exposed by the RM. To this end, an approach called *Q-Learning for RMs (QRM)* was proposed [8]. QRM learns one q-function $\tilde{q}_u$ (i.e., policy) per RM state $u \in U$. Then, given any sample transition, the RM can be used to emulate how much reward each q-value would receive from every RM state. Formally, experience $e = (o,a,o')$ is transformed into a valid experience $(\langle o,u\rangle, a, \langle o',u'\rangle, r)$ for training $\tilde{q}_u$ for each $u \in U$, where $u' = \delta_u(u, L(e))$ and $r = \delta_r(u, L(e))$. Hence, any off-policy learning method can take advantage of these "synthetically" generated experiences to train every q-function $\tilde{q}_u$. When q-learning is used to learn each policy, QRM is guaranteed to converge to an optimal policy when the problem is fully-observable.

To apply QRM on a learned RM in a partially observable environment, we must first learn values for the RM's reward function δ_r from the set of training traces $\mathcal{T}$. We do so by setting $\delta_r(u,l)$ as its empirical expectation over $\mathcal{T}$. In addition, we must handle an issue related to importance sampling. An experience (o,a,o') might be more or less likely depending on the RM state that the agent was in when the experience was collected. For example, experience (o,a,o') might be possible in one RM state u_i but not in u_j. Updating the policy for u_j using (o,a,o') would then introduce an unwanted bias to $\tilde{q}_{u_j}$. We handle this issue by only "transferring" an experience (o,a,o') from u_i to u_j, if the current RM indicates that experience is possible in u_j. For example, if some experience in Figure 1c consists of entering the red room and seeing only one key, then this experience will not be used to update the policies for states u_2, u_3, u_4, and u_6 of the perfect RM in Figure 1d. While this approach will not address the problem in all environments, we leave that as future work.

Simultaneously Learning a Reward Machine and a Policy

We now describe our overall approach for simultaneously finding an RM and exploiting that RM to learn a policy. Our approach starts by collecting a training set of traces $\mathcal{T}$ generated by a random policy during t_w "warmup" steps. This set of traces is used to find an initial RM $\mathcal{R}$ using Tabu search. The algorithm then initializes policy π, sets the RM state to the initial state u_0, and sets the current label l to the initial abstract observation $L(\emptyset,\emptyset,o)$. The standard RL learning loop is then followed: an action a is selected following $\pi(o,x)$, and the agent receives the next observation o' and the immediate reward r. The RM state is then updated to $x' = \delta_u(x, L(o,a,o'))$ and the policy π is improved using whatever RL method is being deployed using the last experience $(\langle o,x\rangle, a, r, \langle o',x'\rangle)$. Note that in an episodic task, the environment and RM are reset whenever a terminal state is reached.

If on any step, there is evidence that the current RM might not be the best one, our approach will attempt to find a new one. Recall that the RM $\mathcal{R}$ was selected using the cardinality of its prediction sets N (1). Hence, if the current abstract observation l' is not in $N_{x,l}$, adding the current trace to $\mathcal{T}$ will increase the size of $N_{x,l}$ for $\mathcal{R}$. As such, the cost of $\mathcal{R}$ will increase and it may no longer be the best RM. Thus, if $l' \notin N_{x,l}$, we add the current trace to $\mathcal{T}$ and search for a new RM. Recall that we use Tabu search, though any discrete optimization method could be applied. Our method only uses the new RM if its cost is lower than $\mathcal{R}$'s. If the RM is updated, a new policy is learned from scratch.

4 Evaluation and Discussion

We tested our approach on three partially observable grid domains, each with the same layout of three rooms with a connecting hallway. The agent can move in the four cardinal directions and can only see what is in the current room. These are stochastic domains where the outcome of an action randomly changes with a 5% probability. The first environment is

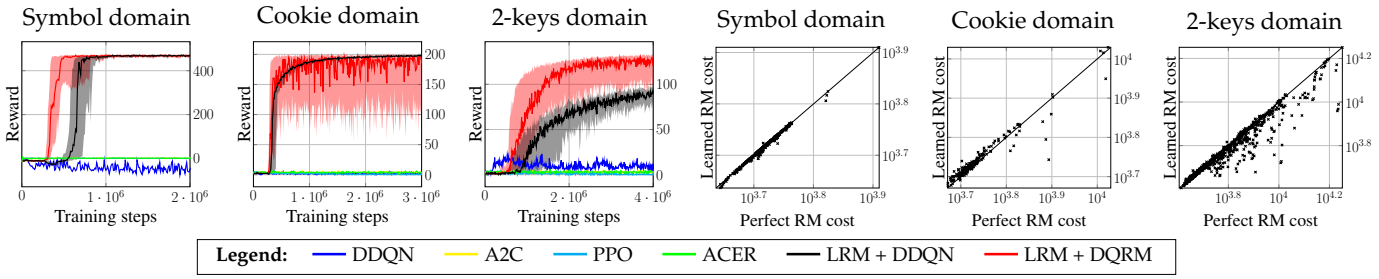


Figure 2: **Left**: Total reward collected every $10,000$ training steps. It shows the median performance over 30 runs and percentile 25 to 75 in the shadowed area for LRM approaches. The maximum performance is reported for the other approaches. **Right**: Comparison between the cost of the perfect RM and the cost of RMs found by Tabu search.

the *symbol domain* (Figure 1a). It has three symbols ♣, ♠, and ♦ in the red and blue rooms. One symbol from {♣, ♠, ♦} and possibly an up or down arrow are randomly placed at the yellow room. Intuitively, that symbol and arrow tell the agent where to go (e.g., ♣ and ↑ tell the agent to go to ♣ in the north room). If there is no arrow, the agent can go to the target symbol in either room. An episode ends when the agent reaches any symbol in the red or blue room, at which point they receive a reward of $+1$ if they reached the correct symbol and -1 for an incorrect symbol. The second environment is the *cookie domain* (Figure 1b). It has a button in the yellow room that, when pressed, makes a cookie randomly appear in the red or blue room. The agent receives reward $+1$ for reaching the cookie and may then go back to the button to make another one appear. Each episode is $5,000$ steps long, during which the agent should attempt to get as many cookies as possible. The final environment is the *2-keys domain* (Figure 1c) that was described in Section 1.

We tested two versions of our Learned Reward Machine (LRM) approach: LRM+DDQN and LRM+DQRM. Both learn an RM from experience as described in Section 3, but LRM+DDQN learns a policy using DDQN [9] while LRM+DQRM uses the modified version of QRM. In all domains, we used $u_{\max} = 10$, $t_w = 200,000$, an epsilon greedy policy with $\epsilon = 0.1$, and a discount factor $\gamma = 0.9$. The size of the Tabu list and the number of steps that the Tabu search performs before returning the best RM found is 100. We compared against 4 baselines: DDQN [9], A2C [5], ACER [10], and PPO [7]. To provide DDQN some memory, its input is set as the concatenation of the last 10 observations, as commonly done by Atari playing agents. In contrast, A2C, ACER, and PPO already use an LSTM to summarize the observation history.

The left side of Figure 2 shows the total reward that each approach gets every $10,000$ training steps. The figure shows that the LRM approaches largely outperform all the baselines. We also note that LRM+DQRM learns faster than LRM+DDQN, but is more unstable. In particular, LRM+DQRM converged to a considerably better policy than LRM+DDQN in the 2-keys domain. We believe this is due to QRM's experience sharing mechanism that allows for propagating sparse reward backwards faster. In contrast, all the baselines outperformed a random policy, but none make much progress on any of the domains, even when run much longer ($5,000,000$ steps).

A key factor in the strong performance of the LRM approaches is that Tabu search finds high-quality RMs in less than 100 search steps (Figure 2, right side). In each plot, a point compares the cost of a handcrafted perfect RM with that of an RM $\mathcal{R}$ that was found by Tabu search while running our LRM approaches, where the costs are evaluated relative to the training set used to find $\mathcal{R}$. Being on or under the diagonal line (as in most of the points in the figure) means that Tabu search is finding RMs whose values are at least as good as the handcrafted RM. Hence, Tabu search is either finding perfect reward machines or discovering that our training set is incomplete and our agent will eventually fill those gaps.

For future work, we plan on exploring the use of recurrent neural networks for finding policies for each RM subtask. Doing so would mean that we might not have to find a perfectly Markovian high-level decomposition. We expect this will allow us to solve problems with less informative labelling functions, and using RMs with fewer states.

References

[1] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, pages 1023–1028, 1994.

[2] F. Glover and M. Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[3] M. Mahmud. Constructing states for reinforcement learning. In *ICML*, pages 727–734, 2010.

[4] N. Meuleau, L. Peshkin, K.-E. Kim, and L. P. Kaelbling. Learning finite-state controllers for partially observable environments. In *UAI*, pages 427–436, 1999.

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937, 2016.

[6] J. Oh, V. Chockalingam, S. Singh, and H. Lee. Control of memory, active perception, and action in minecraft. In *ICML*, pages 2790–2799, 2016.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[8] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *ICML*, pages 2112–2121, 2018.

[9] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016.

[10] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

Value Preserving State-Action Abstractions

David Abel
Department of Computer Science
Brown University
david_abel@brown.edu

Nate Umbanhowar
Department of Computer Science
Brown University
umbanhowar@brown.edu

Khimya Khetarpal
Department of Computer Science
McGill University
khimya.khetarpal@mail.mcgill.ca

Dilip Arumugam
Department of Computer Science
Stanford University
dilip@stanford.edu

Doina Precup
Department of Computer Science
McGill University
dprecup@cs.mcgill.ca

Michael L. Littman
Department of Computer Science
Brown University
mlittman@cs.brown.edu

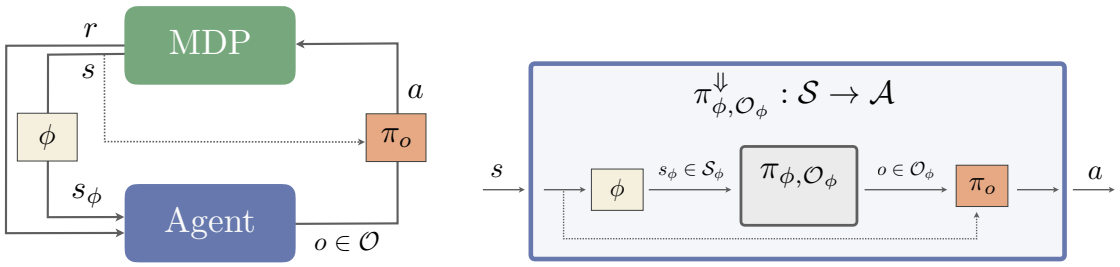
Abstract

We here introduce combinations of state abstractions and options that preserve representation of near-optimal policies. We define ϕ-relative options, a general formalism for analyzing the value loss of options paired with a state abstraction, and prove that there exist classes of ϕ-relative options that preserve near-optimal behavior in any MDP.

Keywords: State Abstraction, Options, Hierarchical Reinforcement Learning

Acknowledgements

We would like to thank Silviu Pitis and Philip Amortila for helpful discussions. This work was supported by a grant from the DARPA L2M program and an ONR MURI grant.



(a) State and Action Abstraction in RL.

(b) Forming a ground policy from a policy over abstract states and ϕ-relative options.

Figure 1: Reinforcement Learning with state abstraction and options: (a) an augmentation of the traditional RL loop wherein an agent reasons in terms of abstract states and chooses among options, and (b) the process for inducing $\pi^{\Downarrow}_{\phi,\mathcal{O}_\phi}$, a Markov policy in the ground MDP, from a $(\phi, \mathcal{O}_\phi, \pi_{\phi,\mathcal{O}_\phi})$ triple.

1 Introduction

We here explore the role of *state* and *action* abstractions in the context of Reinforcement Learning (RL) [24], as pictured in Figure 1a. Our main objective is to clarify which combinations of state and action abstractions support representation of near-optimal policies in Markov Decision Processes (MDPs) [22].

A state abstraction defines an aggregation function that translates the environmental state space $\mathcal{S}$ into $\mathcal{S}_\phi$, where usually $|\mathcal{S}_\phi| \ll |\mathcal{S}|$. With a smaller state space, learning algorithms can learn with less computation, space, and even samples [10, 3, 23, 13, 14]. However, throwing away information about the state space might destroy representation of good policies. An important direction for research is to clarify which state abstractions can preserve near-optimal behavior [9, 17, 14, 1, 2].

We take an action abstraction to be a replacement of the actions of an MDP, $\mathcal{A}$, with a set of options [25], $\mathcal{O}$, which encode long-horizon sequences of actions. Options are known to aid in transfer [16, 6, 28], encourage better exploration [4, 12, 18, 27], and make planning more efficient [20, 21].

The primary contribution of this work introduces combinations of state abstractions and options that preserve representation of near-optimal behavior. We define ϕ-relative options, a general formalism for analyzing the value loss of pairs $(\phi, \mathcal{O}_\phi)$, and prove there are classes of ϕ-relative options that preserve near-optimal behavior in any MDP.

1.1 Background

We first provide brief background on state abstractions and options.

Definition 1 (State Abstraction): *A state abstraction $\phi : \mathcal{S} \to \mathcal{S}_\phi$ maps each ground state, $s \in \mathcal{S}$ into an abstract state, $s_\phi \in \mathcal{S}_\phi$. We denote policies over abstract states as π_ϕ, defined as a mapping $\mathcal{S}_\phi \to \mathcal{A}$.*

Critically, a policy over abstract states induces a unique policy over ground states:

Remark 1. *Any deterministic policy defined over abstract states, $\pi_\phi : \mathcal{S}_\phi \to \mathcal{A}$ induces a unique policy in the original MDP. We denote this ground policy as $\pi^{\Downarrow}_\phi(s)$, and the space of all policies representable in this manner as $\Pi^{\Downarrow}_\phi$.*

For each $s \in \mathcal{S}$, we may pass it through the abstraction to yield $s_\phi = \phi(s)$. To specify an action, we then query $\pi_\phi(s_\phi)$. Using this mapping process we can evaluate a given abstract policy, π_ϕ, by the value of its induced ground policy, $\pi^{\Downarrow}_\phi$. We now define the sub-optimality induced by a given state abstraction ϕ.

Definition 2 (ϕ-Value Loss): *The value loss associated with a state abstraction ϕ denotes the degree of sub-optimality attained by applying the best abstract policy. More formally:*

$$L(\phi) := \min_{\pi^{\Downarrow}_\phi \in \Pi^{\Downarrow}_\phi} \max_{s \in \mathcal{S}} V^*(s) - V^{\pi^{\Downarrow}_\phi}(s). \tag{1}$$

Next we introduce options, a popular formalism for empowering the action space of an agent.

Definition 3 (Option [25]): *An option $o \in \mathcal{O}$ is a triple $\langle \mathcal{I}_o, \beta_o, \pi_o \rangle$, where $\mathcal{I}_o \subseteq \mathcal{S}$ is a subset of the state space denoting where the option initiates; $\beta_o \subseteq \mathcal{S}$, is a subset of the state space denoting where the option terminates; and $\pi_o : \mathcal{S} \to \mathcal{A}$ is a deterministic policy prescribed by the option o.*

Options define abstract actions; the three components indicate where the option o can be executed ($\mathcal{I}_o$), where the option finishes (β_o), and what to do in between these two conditions (π_o).

2 State-Action Abstractions

Together, state and action abstractions can distill complex problems into simple ones [15, 8, 5]. Our treatment of state-action abstraction is related to generating options from a bisimulation metric [11] as proposed by Castro and Precup [7], but distinct from state-action homomorphisms, as explored by Ravindran [23], Taylor et al. [26] and Majeed and Hutter [19]. We here introduce a novel means of combining state abstractions with options, defined as follows:

Definition 4 (ϕ-Relative Option): *For a given ϕ, an option is said to be ϕ-relative if and only if there is some $s_\phi \in \mathcal{S}_\phi$ such that, for all $s \in \mathcal{S}$:*

$$\mathcal{I}_o(s) \equiv s \in s_\phi, \qquad \beta_o(s) \equiv s \notin s_\phi, \qquad \pi_o \in \Pi_{s_\phi}, \tag{2}$$

where $\Pi_{s_\phi} : \{s \mid \phi(s) = s_\phi\} \to \mathcal{A}$ is the set of ground policies defined over states in s_ϕ. We denote $\mathcal{O}_\phi$ as any non-empty set that 1) contains only ϕ-relative options, and 2) contains at least one option that initiates in each $s_\phi \in \mathcal{S}_\phi$.

Intuitively, this means we define options that initiate in each abstract state and terminate once the option leaves the abstract state. For example, in the classical Four Rooms domain, if the state abstraction turns each room into an abstract state, then any ϕ-relative option in this domain would be one that initiates anywhere in one of the rooms and terminates as soon as the option leaves that room. This gives us a powerful formalism for seamlessly combining state abstractions and options.

We henceforth denote $(\phi, \mathcal{O}_\phi)$ as a state abstraction paired with a set of ϕ-relative options. We first show that, similar to Remark 1, any $(\phi, \mathcal{O}_\phi)$ gives rise to an abstract policy over $\mathcal{S}_\phi$ and $\mathcal{O}_\phi$ that *also* induces a unique policy in the ground MDP (over the entire ground state space). We do not here present proofs due to space constraints.

Theorem 1. *Every deterministic policy defined over abstract states and ϕ-relative options, $\pi_{\phi,\mathcal{O}_\phi} : \mathcal{S}_\phi \to \mathcal{O}_\phi$, induces a unique Markov policy over the original MDP, $\pi^{\Downarrow}_{\phi,\mathcal{O}_\phi} : \mathcal{S} \to \mathcal{A}$. We denote $\Pi^{\Downarrow}_{\phi,\mathcal{O}_\phi}$ as the set of policies in the ground MDP representable by the pair $(\phi, \mathcal{O}_\phi)$ via this mapping.*

This theorem gives us a means of translating a policy over ϕ-relative options into a ground policy over $\mathcal{S}$ and $\mathcal{A}$ (this process is visualized in Figure 1b). Consequently, we can define the value loss associated with a set of options paired with a state abstraction: every $(\phi, \mathcal{O}_\phi)$ pair yields a set of policies in the ground MDP, $\Pi^{\Downarrow}_{\phi,\mathcal{O}_\phi}$. The value loss of $\phi, \mathcal{O}_\phi$, then, is just the value loss of the best policy in this set.

Definition 5 ($(\phi, \mathcal{O}_\phi)$-Value Loss): *The value loss of $(\phi, \mathcal{O}_\phi)$ is the smallest degree of suboptimality achievable:*

$$L(\phi, \mathcal{O}_\phi) := \min_{\pi^{\Downarrow}_{\phi,\mathcal{O}_\phi} \in \Pi^{\Downarrow}_{\phi,\mathcal{O}_\phi}} \max_{s \in \mathcal{S}} V^*(s) - V^{\pi^{\Downarrow}_{\phi,\mathcal{O}_\phi}}(s). \tag{3}$$

To characterize the loss of various options, we require a final definition that clarifies what is meant by an option class. We adopt a new formalism that characterizes sets of options as containing *representative* options, defined as follows.

Definition 6 (Option Class): *Let $\mathcal{O}^{all}_\phi$ denote the set of all possible ϕ relative options for a given ϕ. For every s_ϕ, consider a two-place predicate on options of this set, $p_{s_\phi} : \mathcal{O}^{all}_\phi \times \mathcal{O}^{all}_\phi \to \{0, 1\}$. A set of ϕ-relative options is said to belong to the class defined by p_{s_ϕ}, which we denote $\mathcal{O}_{\phi,\mathrm{p}}$, if and only if:*

$$\forall_{s_\phi \in \mathcal{S}_\phi} \forall_{o_1 \in \mathcal{O}^{all}_\phi} \exists_{o_2 \in \mathcal{O}_{\phi,p}} : p_{s_\phi}(o_1, o_2). \tag{4}$$

Intuitively, a class of options consists of choosing a small set of *representative* options from the set of all possible options, where the other options that the representatives are intended to account for are defined by the predicate. In the trivial case, the predicate defines equivalence; if the two options are the same, it is true. In this case, we just recover the set of all options. Instead, we might describe a class of options as those that transition to the same next abstract state from the given s_ϕ; then, we need only retain *one* such option to adhere to this class. Shortly, we will define two classes that possess desirable theoretical properties.

With our definitions in place, we now pose the central question of this work:

> **Central Question:** Are there classes of options that, when paired with well-behaved state abstractions (that is, $L(\phi) = \varepsilon_\phi$), yield a relatively small $L(\phi, \mathcal{O}_\phi)$?

Our main result answers this question in the affirmative; the following two option classes preserve near-optimality. The option classes we introduce guarantee ε closeness of values or models, building off of state abstraction classes from prior work [9, 17, 1]. More concretely:

1. *Similar Q^*-Functions* ($\mathcal{O}_{\phi,Q^*_\varepsilon}$): The ε-similar Q^* predicate defines an option class where, for all s_ϕ:
$$p_{s_\phi}(o_1, o_2) \equiv \max_{s \in \mathcal{S}} |Q^*_{s_\phi}(s, o_1) - Q^*_{s_\phi}(s, o_2)| \leq \varepsilon_Q, \tag{5}$$
where:
$$Q^*_{s_\phi}(s, o) := R(s, \pi_o(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s, \pi_o(s)) \left(\mathbb{1}(s' \in s_\phi) Q^*_{s_\phi}(s', \pi_o) + \mathbb{1}(s' \notin s_\phi) V^*(s') \right). \tag{6}$$
2. *Similar Models* ($\mathcal{O}_{\phi,M_\varepsilon}$): The ε-similar T and R predicate defines an option class where, for all s_ϕ:
$$p_{s_\phi}(o_1, o_2) \equiv ||T^{s'}_{s,o_1} - T^{s'}_{s,o_2}||_\infty \leq \varepsilon_T \; AND \; ||R_{s,o_1} - R_{s,o_2}||_\infty \leq \varepsilon_R, \tag{7}$$
where $R_{s,o}$ and $T^{s'}_{s,o}$ are shorthand for the reward model and multi-time model of Sutton et al. [25].

Our main result establishes the bounded value loss of these two classes.

Theorem 2. *(Main Result) For any ϕ such that $L(\phi) \leq \varepsilon_\phi$, the two introduced classes of ϕ-relative options satisfy:*
$$L(\phi, \mathcal{O}_{\phi,Q^*_\varepsilon}) \leq \varepsilon_\phi + \frac{\varepsilon_Q}{1-\gamma}, \qquad L(\phi, \mathcal{O}_{\phi,M_\varepsilon}) \leq \varepsilon_\phi + \frac{\varepsilon_R + |\mathcal{S}|\varepsilon_T \text{VMAX}}{1-\gamma}. \tag{8}$$

2.1 Discussion

We introduce ϕ-relative options, a simple but expressive formalism for combining state aggregation functions with options. These offer analysis of the quantity $L(\phi, \mathcal{O}_\phi)$, a coherent notion of value loss extended to capture the value loss of joint state-action abstractions. We introduce two classes of ϕ-relative options that are guaranteed to preserve near-optimality in any MDP. We take this to serve as a concrete path toward principled option discovery and use; our main direction for future work is to develop a practical option discovery algorithm that 1) offers synergy with state abstraction, and 2) is guaranteed to retain near-optimal behavior.

References

[1] David Abel, D. Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. In *ICML*, pages 2915–2923, 2016.

[2] David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L. Littman, and Lawson L.S. Wong. State abstraction as compression in apprenticeship learning. In *AAAI*, 2019.

[3] David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *AAAI*, pages 119–125, 2002.

[4] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2017.

[5] Aijun Bai and Stuart Russell. Efficient reinforcement learning with hierarchies of machines by leveraging internal transitions. In *IJCAI*, 2017.

[6] Emma Brunskill and Lihong Li. PAC-inspired option discovery in lifelong reinforcement learning. In *ICML*, pages 316–324, 2014.

[7] Pablo Samuel Castro and Doina Precup. Automatic construction of temporally extended actions for MDPs using bisimulation metrics. In *EWRL*, 2011.

[8] Kamil Ciosek and David Silver. Value iteration with options and state aggregation. *arXiv:1501.03959*, 2015.

[9] Thomas Dean and Robert Givan. Model minimization in Markov decision processes. In *AAAI*, 1997.

[10] Thomas G Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 2000.

[11] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *UAI*, 2004.

[12] Ronan Fruit and Alessandro Lazaric. Exploration–exploitation in MDPs with options. *AISTATS*, 2017.

[13] Jesse Hostetler, Alan Fern, and Tom Dietterich. State aggregation in MCTS. In *AAAI*, 2014.

[14] Nan Jiang, Alex Kulesza, and Satinder Singh. Abstraction selection in model-based reinforcement learning. In *ICML*, pages 179–188, 2015.

[15] Anders Jonsson and Andrew G Barto. Automated state abstraction for options using the U-tree algorithm. In *NeurIPS*, pages 1054–1060, 2001.

[16] George Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, 2007.

[17] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for MDPs. In *ISAIM*, 2006.

[18] Marlos C Machado, Marc G Bellemare, and Michael Bowling. A Laplacian framework for option discovery in reinforcement learning. In *ICML*, 2018.

[19] Sultan Javed Majeed and Marcus Hutter. Performance guarantees for homomorphisms beyond Markov decision processes. *AAAI*, 2019.

[20] Timothy Mann and Shie Mannor. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *ICML*, pages 127–135, 2014.

[21] Timothy A Mann, Shie Mannor, and Doina Precup. Approximate value iteration with temporally extended actions. *Journal of Artificial Intelligence Research*, 2015.

[22] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[23] Balaraman Ravindran. SMDP homomorphisms: An algebraic approach to abstraction in semi Markov decision processes. 2003.

[24] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[25] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

[26] Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate MDP homomorphisms. In *NeurIPS*, 2008.

[27] Saket Tiwari and Philip S Thomas. Natural option critic. *AAAI*, 2019.

[28] Nicholay Topin, Nicholas Haltmeyer, Shawn Squire, John Winder, James MacGlashan, et al. Portable option discovery for automated learning transfer in object-oriented Markov decision processes. In *IJCAI*, 2015.

Bandits with Temporal Stochastic Constraints

Priyank Agrawal*
Indian Institute of Science
Bangalore, India
priyank.93.agrawal@gmail.com

Theja Tulabandhula
University of Illinois at Chicago
Chicago, IL 60607
theja@uic.edu

Abstract

We study the effect of impairment on stochastic multi-armed bandits and develop new ways to mitigate it. Impairment effect is the phenomena where an agent only accrues reward for an action if they have played it at least a few times in the recent past. It is practically motivated by repetition and recency effects in domains such as advertising (here consumer behavior may require repeat actions by advertisers) and vocational training (here actions are complex skills that can only be mastered with repetition to get a payoff). Impairment can be naturally modelled as a temporal constraint on the strategy space, we provide a learning algorithm that achieves sublinear regret. Our regret bounds explicitly capture the cost of impairment and show that it scales (sub-)linearly with the degree of impairment. Beyond the primary objective of calculating theoretical regret guarantees, we also provide experimental evidence supporting our claims.

In Summary, our contributions are three-folds: Modeling arm pull history dependent impairment effect; designing a sublinear regret learning algorithm and showing its relevance in the past literature of reward corruption and delay and finally, supporting our theoretical guarantees with experimental validation.

Keywords: Multi-armed-bandits, Stochastic Impairments, Stochastic History, Doob's Martingale

*Additional details can be found in the full version of this work, *Bandits with Temporal Stochastic Constraints* available at https://arxiv.org/abs/1811.09026

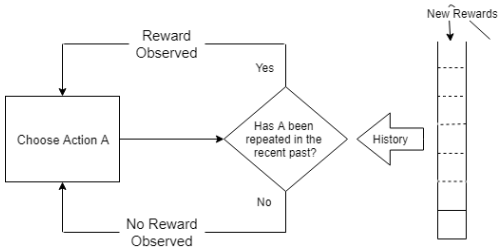


Figure 1: Bandit learning with stochastic impairment.

1 Introduction

In the space of advertising and consumer behavior models, repetition effect [Machleit and Wilson, 1988, Campbell and Keller, 2003] has been well studied, under which, an advertiser's payoff (for instance, click through rate) depends on how frequently they have presented the same ad to the same audience in the recent past. If the advertiser presents a specific ad sporadically, then the aggregated payoff is much lower. If this ad is the best among a collection of ads, then the advertiser will not be able to deduce this from their observations. Further, different ads may need different levels of repetition to obtain payoffs, and this may not be known a priori to the advertiser. This phenomenon also translates to recommendations, such as for products and movies, where repeated display of item(s) can cause positive reinforcement to build over time, culminating in a conversion. And since conversions depend on the past recommendations, they interfere with learning the true underlying (mean) payoffs of different recommendations. In the domain of skill acquisition [DeKeyser, 2007], especially those which are complex [Bosse et al., 2015], a learner may have to repeatedly try each action several times to advance or acquire a reward. Further, they may have to repeat these actions frequently enough so as to not lose the acquired skill [Kang, 2016].

Motivated by the above discussion, we define a new class of problems, which we call *bandit learning with stochastic impairment*, to address the need for repetitions. The defining characteristic here is that a learning algorithm can only accrue rewards if it has played the same arm *enough* number of times in the recent past. The amount by which the algorithm needs to replay the same arm is a measure of impairment. The reward for playing that arm at the current time is instantaneous. A diagram illustrating this is shown in Figure 1.

As the impairment effect is based on the history of arm pulls which is stochastic, usual multi-armed-bandits (MAB) algorithms such as UCB1, SE, MOSS or Thompson Sampling are ineffective because one cannot directly control the number of times an arm is played in a given time window. In fact, if we have an instance in which a couple of arms have almost equal mean payoffs, then the aforementioned algorithms may switch between these very frequently (see Section 4), potentially causing linear regret. We also observe that *impairment* setting is closely related to delay [Pike-Burke et al., 2018, Cesa-Bianchi et al., 2018] and corruption [Lykouris et al., 2018, Gajane et al., 2017] in the reward accrual process. As an unifying view, in all three works one can assume that there is a intermediate function that allows an MAB algorithm to accrue some transformation of the rewards instead of the rewards themselves.

Owing to the nature of impairment effect, learning algorithms necessarily have to ensure that the arms still under consideration are played frequently, perhaps in batches. In particular, if one plays an arm for a sufficiently long period of time, then they can ensure that the instantaneous rewards are always accrued (except some at the beginning). Phase based algorithms are a natural choice given these considerations. This family of algorithms date back to [Agrawal et al., 1988], who considered arm switching costs. To address these issues, we develop UCB-REVISITED+, which expands on the phase-based algorithmic template to mitigate impairments in reward accrual. In particular, UCB-REVISITED+ is based on UCB-REVISITED [Auer and Ortner, 2010] and works under the setting when the expected impairment effect is known. Naturally, the algorithm also works when the impairment is deterministic.

To analyze regret upper bounds for our proposed algorithm, UCB-REVISITED+, we rely on its phased operation to distill reward sequences for each of its arm. We first consider zero-mean sequences by subtracting the mean from each reward, then we segregate portions of the sequence corresponding to the impairment effect. Further, we construct a version of Doob's martingales with the associated filtration set being the entire history of rewards and arm pulls on these sequences. Following the analysis techniques of [Auer and Ortner, 2010] and [Pike-Burke et al., 2018],where the latter uses Freedman's inequality and the Azuma-Hoeffding inequality based Doob's optimal skipping theorem (see Section 3), we finally calculate high probability bounds on the regret and show that the influence of impairment is at best only additive. While our analysis partially overlaps with these two previous works, the random variables related to impairment in our setting do need a qualitatively different treatment.

2 Problem Definition

There are $K > 1$ arms in the set $\mathcal{K}$. Each arm $j \in \mathcal{K}$ is associated with a reward distribution ξ_j, which has support $[0, 1]$. The mean reward for arm j is μ_j. μ^* is the maximum of all μ_j and corresponds to the arm j^*. Define $\Delta_j = \mu^* - \mu_j$.

One of the key aspects of this work is a novel modeling of *impairment* that temporally correlates the rewards accrued by any algorithm with its past sequence of actions. Let $R_{t,j}$ be the reward that would be generated if the j^{th} arm is played at time t. Further, let $J_t \in \mathcal{K}$ represents the arm that is played at time t. Each arm j is associated with an i.i.d. stochastic process (*impairment process*) $\{d_{t,j}\}$ that controls reward accrual in the following way: the learner can only observe the reward $R_{t,j}$ if the arm j was pulled at least $d_{t,j}$ times in the N-most recent time steps, as:

$$X_{t,j} = R_{t,j}\mathbb{I}\left[\left(\sum_{k=\max(t-N,0)}^{t} \mathbb{I}[J_k = j]\right) \geq d_{t,j}\right],$$

where $\mathbb{I}[\,]$ is the indicator function. For simplicity, we demonstrate our solution assuming the mean of each random variable $d_{t,j}$ is equal to $\mathbb{E}[d]$. The tuple $\langle \mathcal{K}, \xi_j, \{d_{t,j}\}, N\rangle$ completely defines a problem instance.

With this context, the learning goal is to design an online algorithm that minimizes (pseudo-) regret for a given time horizon T, as:

$$R_T = \max_{k\in[K]} \mathbb{E}\left[\sum_{t=1}^{T} X_{t,k}\right] - \mathbb{E}\left[\sum_{t=1}^{T} X_{t,J_t}\right]. \tag{1}$$

Unlike the standard setting, the above terms cannot be further simplified because here the accrued rewards, $\{X_{t,j}\}$ depends both on the reward distribution and stochastic history of the arm pulls.

3 Algorithm and Analysis

Notation: Let m index phases. Let $T_j(m)$ refer to the collection of times when the j^{th} arm is played up to phase m. X_t is the reward accrued at time t and $R_{t,j}$ is the reward observed for playing arm j at time t. The sequence of parameters $\{n_m | m = 0, 1, 2, ...\}$ determine the number of consecutive rounds each active arm is played in the phase m, where active arms belong to the set $\mathcal{K}_m$. The estimated mean reward for arm j at the end of phase m is denoted by $\overline{X}_{m,j}$.

In the m_{th} phase, every arm in the set of *active arms* is played consecutively for a certain number of times, say n_m times. We creatively design n_m such that the confidence gap ($\tilde{\Delta}_m$) of the active arms decreases exponentially with each phase, while eliminating the arms with $\Delta_i/2 > \tilde{\Delta}_m$. Our main idea is that the intelligently designed repetition strategy helps to negate impairment effects. As n_m also depends on the impairment statistics, $\mathbb{E}[d]$, we can also incorporate arm specific impairment parameters by considering a suitable $n_{m,j}$ for the specific arm j. Here, we assume that the algorithm can distinguish between the two possibilities: zero valued rewards and no rewards received due to impairment.

We give a constructive proof for the choice of n_m such that the estimated mean reward for an arm j is atmost $\tilde{\Delta}_m$ from its true mean with high probability.

Lemma 3.1. [For details refer **Lemma 4.3** in Agrawal and Tulabandhula [2018]] There exists a positive n_m for which the estimates $\overline{X}_{m,j}$, calculated by the Algorithm 1 for the active arm j ($j \in K_m$) and phase m, satisfy the following with probability $\geq (1 - \frac{2}{T^2})$:

$$\overline{X}_{m,j} - \mu_j \leq \tilde{\Delta}_m/2.$$

Proof. We provide a proof sketch here. For any active arm j and phase m , let $S_{m,j}$ denote the time in this phase when the algorithm starts playing this arm. Similarly let $U_{m,j}$ denote the time in this phase when the algorithm stops playing this arm. We define a filtration $\{\mathcal{G}_s\}_{s=0}^{\infty}$ by setting $\{\mathcal{G}_0\} = \{\Omega, \phi\}$ and defining $\{\mathcal{G}_t\}$ to be the σ-algebra over $(X_1, X_2....X_t, J_1, J_2....J_t, d_{1,J_1}, d_{2,J_2}....d_{t,J_t}, R_{1,J_1}, R_{2,J_2}...R_{t,J_t})$. Then, it follows that for each arm j:

$$\sum_{i=1}^{m}\sum_{t=S_{i,j}}^{U_{i,j}} (X_t - \mu_j) \leq w_m = \sum_{i=1}^{m}\sum_{t=S_{i,j}}^{U_{i,j}} (R_{t,J_t} - \mu_j) - \sum_{i=1}^{m}\sum_{t=S_{i,j}}^{U_{i,j}} R_{t,J_t}\mathbb{I}\{t \leq S_{i,j} + d_{t,J_t}\}. \tag{2}$$

Define $A_{i,t} := R_{t,J_t}\mathbb{I}\{t \leq S_{i,j} + d_{t,J_t}\}$ and also $M_t := \sum_{i=1}^{m} A_{i,t}\mathbb{I}\{S_{i,j} \leq t \leq U_{i,j}\}$. We rewrite (2) in terms of M_t as:

$$\sum_{i=1}^{m}\sum_{t=S_{i,j}}^{U_{i,j}} (X_t - \mu_j) \leq w_m = \underbrace{\sum_{i=1}^{m}\sum_{t=S_{i,j}}^{U_{i,j}} (R_{t,J_t} - \mu_j)}_{\text{Term 1}} + \underbrace{\sum_{t=1}^{U_{m,j}} (\mathbb{E}[M_t|\mathcal{G}_{t-1}] - M_t)}_{\text{Term 2}} - \underbrace{\sum_{t=1}^{U_{m,j}} \mathbb{E}[M_t|\mathcal{G}_{t-1}]}_{\text{Term 3}}. \tag{3}$$

Input: A set of arms $\mathcal{K}$, time horizon T, and parameters $\{n_m | m = 0, 1, 2, ...\}$.
Initialization: phase index $m = 1$, $\mathcal{K}_m = \mathcal{K}$, $\tilde{\Delta}_1 = 1$, $T_j(m) = \phi \ \ \forall j \in \mathcal{K}$, and time index $t = 1$.
while $t \leq T$ **do**
  Play arms:
  for *each active arm j in $\mathcal{K}_m$* **do**
    Set $T_j(m) = T_j(m-1)$ if $m > 1$.
    Play j for $n_m - n_{m-1}$ consecutive rounds. In each round t:
    if $\left(\sum_{k=\max(t-N,0)}^{t} \mathbb{I}[J_k = j]\right) \geq d_{t,j}$ **then**
      Observe reward $X_{t,j}$ and add t to $T_j(m)$.
    end
  end
  Eliminate Suboptimal Arms:
  for *each active arm j in $\mathcal{K}_m$* **do**
    $\overline{X}_{m,j} = \frac{1}{n_m} \sum_{t \in T_j(m)} X_{t,j}$.
  end
  Construct $\mathcal{K}_{m+1}$ by eliminating arms j in $\mathcal{K}_m$ for which
  $\overline{X}_{m,j} + \tilde{\Delta}_m/2 < \max_{j' \in \mathcal{K}_m} \overline{X}_{m,j'} - \tilde{\Delta}_m/2$.
  Update the confidence bound:
  Set $\tilde{\Delta}_{m+1} = \frac{\tilde{\Delta}_m}{2}$.
  Increment phase index m by 1.
end

Algorithm 1: UCB-REVISITED+

Now we calculate high probability expression for w_m in terms n_m, T and impairment statistics, $\mathbb{E}[d]$. In the **Term 1**, $(R_{t,J_t} - \mu_j)$ is a random variable measurable on the filtration $\{\mathcal{G}_t\}$ with $\mathbb{E}[R_{t,J_t} - \mu_j] = 0$. We apply a version of Azuma-Hoefdding for Doob's Martingales (refer Lemma 4.2 in Agrawal and Tulabandhula [2018] or Lemma 11 in Pike-Burke et al. [2018] for details) to obtain $\sum_{i=1}^{m} \sum_{t=S_{i,j}}^{U_{i,j}} (R_{t,J_t} - \mu_j) \leq \sqrt{n_m \log(T)}$. Secondly, for the **Term 2**, we prove that $\sum_{t=1}^{U_{m,j}} (\mathbb{E}[M_t | G_{t-1}] - M_t)$ is a martingale. Then, we use Bernstein Inequality for Martingales (refer Lemma 4.1 in Agrawal and Tulabandhula [2018] or Theorem 10 in Pike-Burke et al. [2018]) and obtain $\sum_{t=1}^{U_{m,j}} (\mathbb{E}[M_t | G_{t-1}] - M_t) \leq \frac{2}{3}\log(T) + \sqrt{\frac{4\log^2(T)}{9} + 4m\mathbb{E}[d]\log(T)}$ with high probability. **Term 3** has a trivial non-negative upper bound. Finally, we impose the condition $w_m \leq \tilde{\Delta}_m/2$, ensuring the elimination condition of the Algorithm 1 holds good and obtain:

$$n_m \leq 1 + \frac{4\log(T)}{\tilde{\Delta}_m^2} + \frac{16\log(T)}{3\tilde{\Delta}_m} + \frac{8\sqrt{m\mathbb{E}[d]\log(T)}}{\tilde{\Delta}_m}. \tag{4}$$

This completes the proof. □

Having accomplished the difficult task of calculating the phase duration, n_m, we turn our attention to the regret upper bounds which are stated in the following theorem:

Theorem 3.1. [Proof follows that of **Theorem 4.1** in Agrawal and Tulabandhula [2018]] The expected (pseudo-)regret of UCB-REVISITED+ (Algorithm 1) is bounded as:

$$R_T \leq \sum_{i \in K'} \left(\Delta_i + \frac{64\log(T)}{\Delta_i} + \frac{64\log(T)}{3} + 32\sqrt{\log\left(\frac{4}{\Delta_i}\right)\mathbb{E}[d]\log(T)} \right) \quad + \sum_{i \in K' : \Delta_i > \lambda} \frac{2\Delta_i}{T} + \sum_{i \in K'} \frac{32}{T} + \max_{i \in K'' : \Delta_i < \lambda} \Delta_i T,$$

where $K'' = \{i \in K | \Delta_i > 0\}$ and $K' = \{i \in K | \Delta_i > \lambda\}$.

Corollary 3.1. For all $T \geq K$ and choosing $\lambda = \sqrt{\frac{K\log(T)}{T}}$, the expected (pseudo-)regret of UCB-REVISITED+ is upper bounded as:

$$R_T \leq \mathrm{O}\left(\sqrt{KT\log T} + K\sqrt{\log^2 T\mathbb{E}[d]}\right).$$

We note that the effect of impairment only appears as an additive term to the "usual" regret bounds of $\mathrm{O}(\sqrt{KT\log T})$, the magnitude of which depends gracefully on $\sqrt{\mathbb{E}[d]}$. The primary objective of this work is obtain tractable theoretical regret upper bounds for effective learning in impairment setting. In the following section we reinforce our result with relevant simulations.

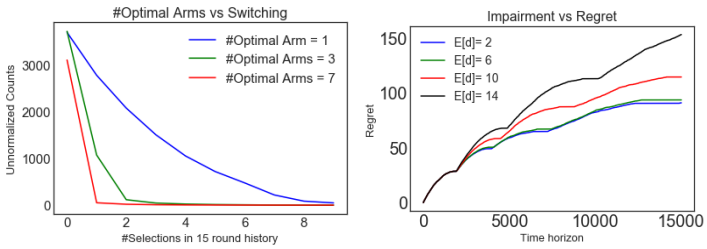


Figure 2: *Left:* Plot of unnormalized counts versus number of same arm plays in the past 15 rounds by UCB1 for three different settings. *Right:* Performance (cumulative regret) of UCB-REVISITED+ as the impairment level is varied.

4 Experiments and Conclusion

We use a simple setup of $K = 30$ arms and set the reward distributions to be the Bernoulli with randomly chosen biases. The horizon length $T = 5000$. We then run UCB1 under three different configurations (1, 3 and 7 optimal arms respectively). Figure 2(left) shows the unnormalized counts of *same arm plays* in the past 15 plays. This was computed by checking how many times the current arm was also played in the past 15 rounds. As expected, as the number of optimal arms increases, the counts of same arm plays decreases rapidly. This indicates that UCB1 and other related algorithms may perform poorly in settings with impairment.

We also show the performance of UCB-REVISITED+ (Algorithm 1) for varying levels of impairment. The number of arms in this experiment is 10. Impairment is stochastic and is simulated using the absolute value normal distribution with means = $\{2, 6, 10, 14\}$ and the standard deviation being proportional to the arm index. The fixed impairment parameter is $N = 20$, and the time horizon is 10000. From Figure 2(right), we can observe that as the cumulative regret increases as $\mathbb{E}[d]$ is increased. It can also be shown that the regret of UCB-REVISITED+ grows as $O(\sqrt{\mathbb{E}[d]})$.

To conclude, we propose a model of impairment and develop new bandit algorithms whose worst case regret depends (sub)linearly on the impairment level. Some future directions include lower bounds and modelling similar impairment setting with Contextual Bandits and MDPs. Also, modeling of contrasting "wearing-out" effects together with the reinforcing ones due to repetition, is an open problem.

References

Priyank Agrawal and Theja Tulabandhula. Bandits with temporal stochastic constraints. *arXiv preprint arXiv:1811.09026*, 2018.

Rajeev Agrawal, MV Hedge, and Demosthenis Teneketzis. Asymptotically efficient adaptive allocation rules for the multiarmed bandit problem with switching cost. *IEEE Transactions on Automatic Control*, 33(10):899–906, 1988.

Peter Auer and Ronald Ortner. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

Hans Martin Bosse, Jonathan Mohr, Beate Buss, Markus Krautter, Peter Weyrich, Wolfgang Herzog, Jana Jünger, and Christoph Nikendei. The benefit of repetitive skills training and frequency of expert feedback in the early acquisition of procedural skills. *BMC medical education*, 15(1):22, 2015.

Margaret C Campbell and Kevin Lane Keller. Brand familiarity and advertising repetition effects. *Journal of consumer research*, 30(2):292–304, 2003.

Nicolo Cesa-Bianchi, Claudio Gentile, and Yishay Mansour. Nonstochastic bandits with composite anonymous feedback. In *Conference on Learning Theory*, pages 750–773, 2018.

Robert DeKeyser. Skill acquisition theory. *Theories in second language acquisition: An introduction*, 97113, 2007.

Pratik Gajane, Tanguy Urvoy, and Emilie Kaufmann. Corrupt bandits for privacy preserving input. *ArXiv preprint arXiv:1708.05033*, 2017.

Sean HK Kang. Spaced repetition promotes efficient and effective learning: Policy implications for instruction. *Policy Insights from the Behavioral and Brain Sciences*, 3(1):12–19, 2016.

Thodoris Lykouris, Vahab Mirrokni, and Renato Paes Leme. Stochastic bandits robust to adversarial corruptions. In *ACM SIGACT Symposium on Theory of Computing*, pages 114–122. ACM, 2018.

Karen A Machleit and R Dale Wilson. Emotional feelings and attitude toward the advertisement: The roles of brand familarity and repetition. *Journal of Advertising*, 17(3):27–35, 1988.

Ciara Pike-Burke, Shipra Agrawal, Csaba Szepesvari, and Steffen Grunewalder. Bandits with delayed, aggregated anonymous feedback. In *International Conference on Machine Learning*, pages 4102–4110, 2018.

Hacking Google reCAPTCHA v3 using Reinforcement Learning

Ismail Akrout*
Télécom ParisTech
akrout.ismail@gmail.com

Amal Feriani*
Ankor AI
amal.feriani@gmail.com

Mohamed Akrout
University of Toronto
makrout@cs.toronto.edu

Abstract

We present a Reinforcement Learning (RL) methodology to bypass Google reCAPTCHA v3. We formulate the problem as a grid world where the agent learns how to move the mouse and click on the reCAPTCHA button to receive a high score. We study the performance of the agent when we vary the cell size of the grid world and show that the performance drops when the agent takes big steps toward the goal. Finally, we use a divide and conquer strategy to defeat the reCAPTCHA system for any grid resolution. Our proposed method achieves a success rate of 97.4% on a 100×100 grid and 96.7% on a 1000×1000 screen resolution.

Keywords: Reinforcement Learning, reCAPTCHA, Security, Artificial Intelligence, Machine Learning

Acknowledgements

We thank Douglas Tweed for his valuable feedback and helpful discussions.

*equal contribution

1 Introduction

Artificial Intelligence (AI) has been experiencing unprecedented success in the recent years thanks to the progress accomplished in Machine Learning (ML), and more specifically Deep Learning (DL). These advances raise several questions about AI safety and ethics [1]. In this work, we do not provide an answer to these questions but we show that AI systems based on ML algorithms such as reCAPTCHA v3 [2] are still vulnerable to automated attacks. Google's reCAPTCHA system, for detecting bots from humans, is the most used defense mechanism in websites. Its purpose is to protect against automated agents and bots, attacks and spams. Previous versions of Google's reCAPTCHA (v1 and v2) present tasks (images, letters, audio) easily solved by humans but challenging for computers. The reCAPTCHA v1 presented a distorted text that the user had to type correctly to pass the test. This version was defeated by Bursztein et al. [3] with 98% accuracy using ML-based system to segment and recognize the text. As a result, image-based and audio-based reCAPTCHAs were introduced as a second version. Researchers have also succeeded in hacking these versions using ML and more specifically DL. For example, the authors in [4] designed an AI-based system called *UnCAPTCHA* to break Google's most challenging audio reCAPTCHAs. On 29 October 2018, the official third version was published [5] and removed any user interface. Google's reCAPTCHA v3 uses ML to return a risk assessment score between 0.0 and 1.0. This score characterize the trustability of the user. A score close to 1.0 means that the user is human.

In this work, we introduce an RL formulation to solve this reCAPTCHA version. Our approach is programmatic: first, we propose a plausible formalization of the problem as a Markov Decision Process (MDP) solvable by state-of-the-art RL algorithms; then, we introduce a new environment for interacting with the reCAPTCHA system; finally, we analyze how the RL agent learns or fails to defeat Google reCAPTCHA. Experiment results show that the RL agent passes the reCAPTCHA test with 97.4 accuracy. To our knowledge, this is the first attempt to defeat the reCAPTCHA v3 using RL .

2 Method

2.1 Preliminaries

An agent interacting with an environment is modeled as a Markov Decision Process (MDP) [6]. A MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, P, r)$ where $\mathcal{S}$ and $\mathcal{A}$ are the sets of possible states and actions respectively. $P(s, a, s^{'})$ is the transition probabilities between states and r is the reward function. Our objective is to find an optimal policy π^* that maximizes the future expected rewards. Policy-based methods directly learn π^*. Let's assume that the policy is parameterized by a set of weights w such as $\pi = \pi(s, w)$. Then, the objective is defined as: $J(w) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \right]$ where γ is the discount factor and r_t is the reward at time t.

Thanks to the policy gradient theorem and the gradient trick [7], the Reinforce algorithm [8] estimates gradients using (1).

$$\nabla \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \right] = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \nabla \log \pi(a_t|s_t) R_t \right] \tag{1}$$

R_t is the future discounted return at time t defined as $R_t = \sum_{k=t}^{T} \gamma^{(k-t)} \cdot r_k$, where T marks the end of an episode.

Usually the equation (1) is formulated as the gradient of a loss function $L(w)$ defined as follows: $L(w) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla \log \pi(a_t^i|s_t^i) R_t^i$ where N is the a number of collected episodes.

2.2 Settings

To pass the reCAPTCHA test, a human user will move his mouse starting from an initial position, perform a sequence of steps until reaching the reCAPTCHA check-box and clicking on it. Depending on this interaction, the reCAPTCHA system will reward the user with a score. In this work, we modeled this process as a MDP where the state space $\mathcal{S}$ is the possible mouse positions on the web page and the action space is $\mathcal{A} = \{up, left, right, down\}$. Using these settings, the task becomes similar to a grid world problem.

As shown in Figure 1, the starting point is the initial mouse position and the goal is the position of the reCAPTCHA is the web page. For each episode, the starting point is randomly chosen from a top right or a top left region representing 2.5% of the browser window's area (5% on the x-Axis and 5% on the y-Axis). A grid is then constructed where each pixel between the initial and final points is a possible position for the mouse. We assume that a normal user will not necessary move the mouse pixel by pixel. Therefore, we defined a cell size c which is the number of pixels between two consecutive positions. For example, if the agent is at the position (x_0, y_0) and takes the action $left$, the next position is then $(x_0 - c, y)$.

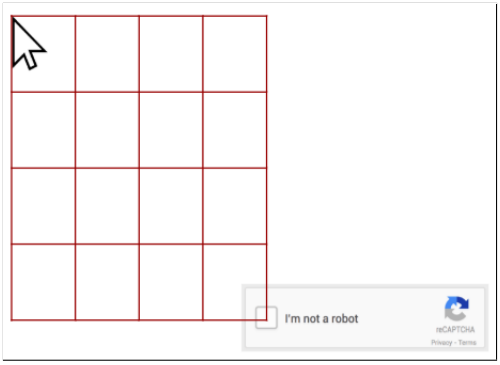


Figure 1: The agent's mouse movement in a MDP

One of our technical contributions consists in our ability to simulate the same user experience as any normal reCAPTCHA user. This was challenging since reCAPTCHA system uses different methods to distinguish fake or headless browsers, inorganic behaviors of the mouse, etc. Our environment overcomes all these problems. For more details about the environment implementation, refer to section 6. At each episode, a browser page will open up with the user mouse at a random position, the agent will take a sequence of actions until reaching the reCAPTCHA or the horizon limit T defined as twice the grid diagonal i.e. $T = 2 \times \sqrt{a^2 + b^2}$ where a and b are the grid's height and width respectively. Once the episode ends, the user will receive the feedback of the reCAPTCHA algorithm as would any normal user.

3 Experiments and Results

We trained a Reinforce agent on a grid world of a specific size. Our approach simply applies the trained policy to choose optimal actions in the reCAPTCHA environment. Our results presented are the success rates across 1000 runs. We consider that the agent successfully defeated the reCAPTCHA if it obtained a score of 0.9. In our experiments, the discount factor was $\gamma = 0.99$. The policy network was a vanilla two fully connected layer network. The parameters were learned with a learning rate of 10^{-3} and a batch size of 2000. Figure 3 shows the results for a 100×100 grid. Our method successfully passed the reCAPTCHA test with a success rate of 97.4%.

Next, we consider testing our method on bigger grid sizes. If we increase the size of the grid, the state space dimension $|\mathcal{S}|$ increases exponentially and it is not feasible to train a Reinforce algorithm with a very high dimensional state space. For example, if we set the grid size to 1000×1000 pixels, the state space becomes 10^6 versus 10^4 for a 100×100. This is another challenge that we address in this paper: how to attack the reCAPTCHA system for different resolutions without training an agent for each resolution?

4 An efficient solution to any grid size

In this section, we propose a divide and conquer technique to defeat the reCAPTCHA system for any grid size without retraining the RL agent. The idea consists in dividing the grid into sub-grids of size 100×100 and then applying our trained agent on these sub-grids to find the optimal strategy for the bigger screen (see Figure 2). Figure 3 shows that this approach is effective and the success rates for the different tested sizes exceed 90%.

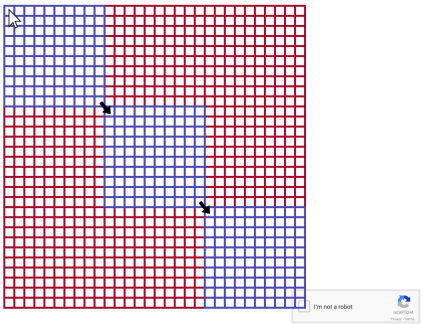


Figure 2: Illustration of the divide and conquer approach: the agent runs sequentially on the diagonal grid worlds in purple. The grid worlds in red are not explored.

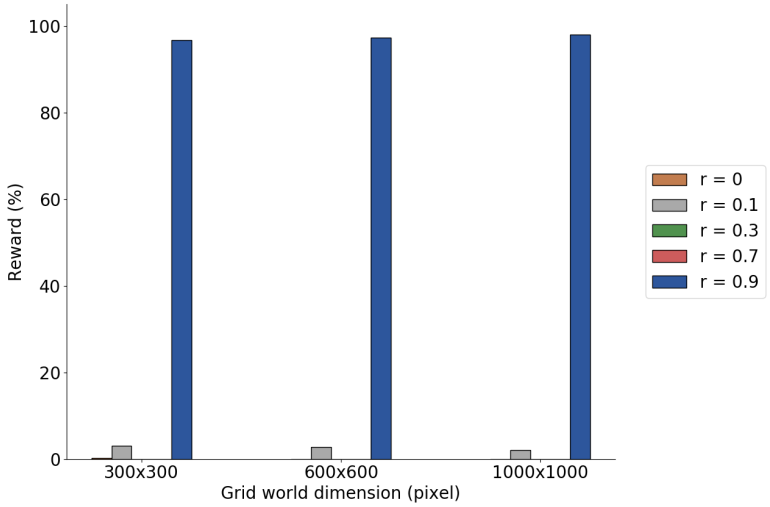
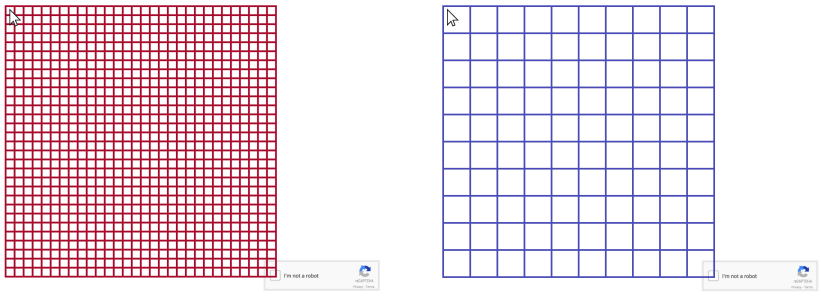


Figure 3: Reward distribution of the RL agent on different grid resolutions over 1000 episodes

5 Effect of cell size

Here, we study the sensitivity of our approach to the cell size as illustrated in Figure 4.



(a) cell size 1x1 pixel (b) cell size 3x3 pixel

Figure 4: Illustration of the effect of the cell size on the state space

Figure 5 illustrates the obtained performance. We observe that when the cell size increases, the success rate of the agent decreases. For, cell size of 10, the RL agent is detected as a bot in more than 20% of the test runs. We believe that this decline is explained by the fact, with a big cell size, the agent scheme will contain more jumps which may be considered as non-human behavior by the reCAPTCHA system.

6 Details of the reCAPTCHA environment

Most previous works (e.g [4]) used the browser automation software *Selenium* [9] to simulate interactions with the re-CAPTCHA system. At the beginning, we adopted the same approach but we observed that the reCAPTCHA system always returned low scores suggesting that the browser was detected as fake. After investigating the headers of the HTTP queries, we found an automated header in the webdriver and some additional variables that are not defined in a normal browser, indicating that the browser is controlled by a script. This was confirmed when we observed that the reCAPTCHA system with *Selenium* and a human user always returns a low score.

It is possible to solve this problem in two different ways. The first consists in creating a proxy to remove the automated header while the second alternative is to launch a browser from the command line and control the mouse using dedicated Python packages such as the *PyAutoGUI* library [10]. We adopted the second option since we cannot control the mouse using *Selenium*. Hence, unlike previous approches, our environment does not use browser automation tools.

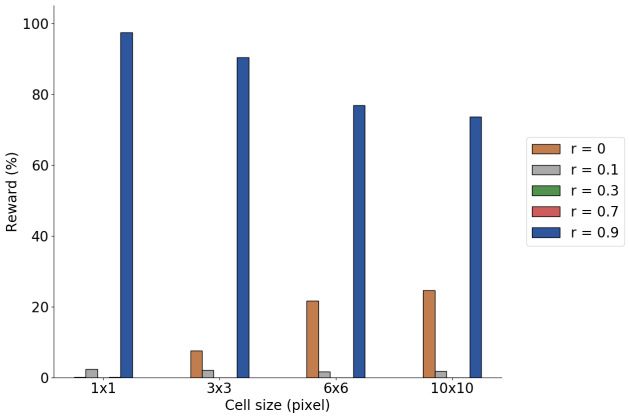


Figure 5: Reward distribution for different cell sizes over 1000 episodes

Another attempt to use *Tor* [11] to change the IP address did not pass the reCAPTCHA test and resulted in low scores (i.e 0.3). It is possible that the reCAPTCHA system uses an API services such as *ExoneraTor* [12] to determine if the IP address is part of the Tor network or not on a specific date.

We also discovered that simulations running on a browser with a connected Google account receive higher scores compared when no Google account is associated to the browser.

To summarize, in order to simulate a human-like experience, our reCAPTCHA environment (1) does not use browser automation tools (2) is not connected using a proxy or VPN (3) is not logged in with a Google account.

7 Conclusion

This paper proposes a RL formulation to successfully defeat the most recent version of Google's reCAPTCHA. The main idea consists in modeling the reCAPTCHA test as finding an optimal path in a grid. We show how our approach achieves more than 90% success rate on various resolutions using a divide and conquer strategy. This paper should be considered as the first attempt to pass the reCAPTCHA test using RL techniques. Next, we will deploy our approach on multiple pages and verify if the reCAPTCHA adaptive risk analysis engine can detect the pattern of attacks more accurately by looking at the activities across different pages on the website.

References

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *CoRR*, 2016.

[2] Google. reCAPTCHA v3's website. https://developers.google.com/recaptcha/docs/v3, 2018. [Online; accessed 15-February-2019].

[3] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John.C Mitchell. The end is nigh: Generic solving of text-based captchas. *USENIX Workshop on Offensive Technologies*, 2014.

[4] Kevin Bock, Daven Patel, George Hughey, and Dave Levin. uncaptcha: A low-resource defeat of recaptcha's audio challenge. *USENIX Workshop on Offensive Technologies*, 2017.

[5] Google. reCAPTCHA v3's official announcement. https://webmasters.googleblog.com/2018/10/introducing-recaptcha-v3-new-way-to.html, 2018. [Online; accessed 15-February-2019].

[6] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[7] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[8] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, May 1992.

[9] Selenium. https://www.seleniumhq.org/. [Online; accessed 15-February-2019].

[10] PyAutoGUI. https://pyautogui.readthedocs.io/en/latest/. [Online; accessed 15-February-2019].

[11] Tor. https://www.torproject.org/. [Online; accessed 15-February-2019].

[12] ExoneraTor. https://metrics.torproject.org/exonerator.html. [Online; accessed 15-February-2019].

DynoPlan: Combining Motion Planning and Deep Neural Network based Controllers for Safe HRL

Daniel Angelov
School of Informatics
University of Edinburgh
d.angelov@ed.ac.uk

Yordan Hristov
School of Informatics
University of Edinburgh
y.hristov@ed.ac.uk

Subramanian Ramamoorthy
School of Informatics
University of Edinburgh
s.ramamoorthy@ed.ac.uk

Abstract

Many realistic robotics tasks are best solved compositionally, through control architectures that sequentially invoke primitives and achieve error correction through the use of loops and conditionals taking the system back to alternative earlier states. Recent end-to-end approaches to task learning attempt to directly learn a single controller that solves an entire task, but this has been difficult for complex control tasks that would have otherwise required a diversity of local primitive moves, and the resulting solutions are also not easy to inspect for plan monitoring purposes. In this work, we aim to bridge the gap between hand designed and learned controllers, by representing each as an option in a hybrid hierarchical Reinforcement Learning framework - DynoPlan. We extend the options framework by adding a dynamics model and the use of a nearness-to-goal heuristic, derived from demonstrations. This translates the optimization of a hierarchical policy controller to a problem of planning with a model predictive controller. By unrolling the dynamics of each option and assessing the expected value of each future state, we can create a simple switching controller for choosing the optimal policy within a constrained time horizon similarly to hill climbing heuristic search. The individual dynamics model allows each option to iterate and be activated independently of the specific underlying instantiation, thus allowing for a mix of motion planning and deep neural network based primitives. We can assess the safety regions of the resulting hybrid controller by investigating the initiation sets of the different options, and also by reasoning about the completeness and performance guarantees of the underpinning motion planners.

Keywords: hierarchical options learning; safe motion planning; dynamics model

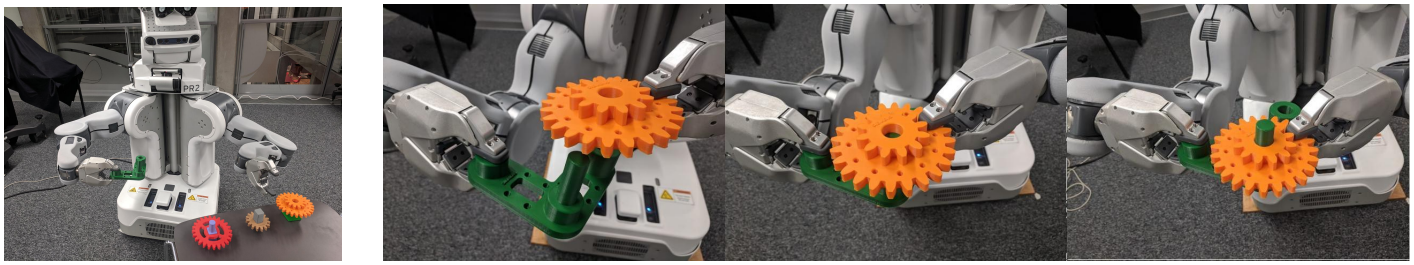
Acknowledgements

This research is supported by the Engineering and Physical Sciences Research Council (EPSRC), as part of the CDT in Robotics and Autonomous Systems at Heriot-Watt University and The University of Edinburgh. Grant reference EP/L016834/1., and by an Alan Turing Institute sponsored project on Safe AI for Surgical Assistance.

1 Introduction

Open world tasks often involve sequential plans. The individual steps in the sequence are usually quite independent from each other, hence can be solved through a number of different methods, such as motion planning approaches for reaching, grasping, picking and placing, or through the use of end-to-end neural network based controllers for a similar variety of tasks. In many practical applications, we wish to combine such a diversity of controllers. This requires them to share a common domain representation. For instance the problem of assembly can be represented as motion planning a mechanical part in proximity to an assembly and subsequently the use of a variety of wiggle policies to fit together the parts. Alternatively, an end-to-end policy can be warm-started by using samples from the motion planner, which informs how to bring the two pieces together and the alignment sub-policy needed, as in [1]. The resulting policy is *robust* in the sense that the task of bringing together the assembly can be achieved from a large set of initial conditions and perturbations.

A hybrid hierarchical control strategy, in this sense, allows for different capabilities to be independently learned and composed into a task solution with multiple sequential steps. We propose a method that allows for these individual steps to consist of commonly used motion planning techniques as well as deep neural network based policies that are represented very differently from their sampling based motion planning counterparts. We rely on these controllers to have a dynamic model of the active part of their state space, and a sense of how close they are to completing the overall task. This allows the options based controller to predict the future using any of the available methods and then determine which one would bring the world state to one closest to achieving the desired solution - in the spirit of model based planning.



(a) Gear Assembly (b) Option 4 of inserting a gear on a peg

Figure 1: The gear assembly problem executed by the robot. The execution of option 4, (Section.5) is shown on the right.

Modern Deep Reinforcement Learning (DRL) approaches focus on generating small policies that solve individual problems (pick up/grasp/push) [2], or longer range end to end solutions illustrated in modern games. Typically, in order to provide a good initialization for the optimization algorithm, expert demonstrations are provided either through human demonstration [3] or through the use of a motion planner as an initial approximation to the solution [1]. In problems that allow for a simulator to be used as part of the inference and learning procedure, DNN & tree based approaches have shown great promise in solving Chess, Go, Poker. To extend these methods to more general domains, a world dynamics model is required to approximate the environment as in [4].

DynoPlan aims at extending the options framework in the following ways:

- We learn a dynamics model $s_{t+1} \sim \mathcal{D}(s_t, a_t)$ for each option that predicts the next state of the world given the current action; and
- We learn a goal heuristic $\mathcal{G}(s_t)$ that gives a distribution as an estimate of how close the state is to completing the task, based on the demonstrations.

This allows for the higher level controller to perform reasoning about sequentially applying controllers in overlapping initiation sets for completing a task.

We aim to show that we can use off-the-shelf model-based controllers in parts of the state space, where their performance is already optimized, and model-free methods for states without correspondingly robust or easily scriptd solutions, combining these two categories of controllers into a hybrid solution.

2 Related Work

Our method sits between learning policies over options as in [5]; and computing solutions using learning from demonstration such as through inverse reinforcement learning [6]. Reinforcement Learning is intrinsically based on the forward search of good states through experience. The update of the quality of an action at a particular state is performed by the iterative application of the Bellman equation. Performing updates in a model-free method must overcome the problems of sparse reward and credit assignment. Introducing a learned model that summarizes the dynamics of the problem can alleviate some scaling issues as in [4]. However, searching for a general world model remains hard and we are not aware

of methods that can achieve the desired performance levels in physical real world tasks. Such problems usually exhibit a hierarchical sequential structure - e.g. the *waking up routine* is a sequence of actions, some of which are conditioned on the previous state of the system.

The options framework provides a formal way to work with hierarchically structured sequences of decisions made by a set of RL controllers. An option consists of a policy $\pi_\omega(a_t|s_t)$, an initiation set $\mathcal{I}$ and termination criteria $\beta_\omega(s_t)$ - probability of terminating the option or reaching the terminal state for the option. A policy over options $\pi_\Omega(\omega_t|s_t)$ is available to select the next option when the previous one terminates as shown by [7, 8].

Temporal abstractions have been extensively researched by [9, 7]. The hierarchical structure helps to simplify the control, allows an observer to disambiguate the state of the agent, and encapsulates a control policy and termination of the policy within a subset of the state space of the problem. This split in the state space allows us to verify the individual controller within the domain of operation - [10], deliberate the cost of an option and increase the interpretability - [11]. Our method borrows this view of temporally abstracting trajectories and extends it by enforcing a dynamics model for each of the options allowing out agent to incorporate hindsight in its actions.

To expedite the learning process, we can provide example solution trajectories by demonstrating solutions to the problem. This can be used to learn safe policies [12]. Alternatively, it can be used to calculate the relative value of each state by Inverse Reinforcement Learning [6]. For instance, we can expect that agents would be approximately rational in achieving their goal, allowing [13] to infer them. Exploring the space of options may force us to consider ones that are unsafe for the agent. [14] rephrases the active inverse reinforcement learning to optimize the agents policy in a risk-aware method. Our work partitions the space of operation of each option, allowing that area to inherit the safety constraints that come associated with the corresponding policy.

3 Problem Definition

We assume there exists an already learned set of options $\mathcal{O} = \{o_1, o_2, ..., o_N\}$ and a set of tasks $\mathcal{K} = \{K_1, K_2, \ldots, K_L\}$. Each option o_ω is independently defined by a policy $\pi_\omega(s) \rightarrow a$, $s \in \mathcal{S}_\omega$, $a \in \mathcal{A}_\omega$, an initiation set $\mathcal{I}_\omega, \mathcal{I}_\omega \subseteq \mathcal{S}_\omega$ where the policy can be started, and a termination criteria β_ω. We extend the options formulation by introducing a forward dynamics model $s_{t+1} \sim \mathcal{D}_\omega(s_t)$, which is a stochastic mapping, and a goal metric $g \sim \mathcal{G}_{K_j}(s_t), 0 \leq g \leq 1$, that estimates the progress of the state s_t with respect to the desired world configuration. We aim for $\mathcal{G}_{K_j}$ to change monotonically through the demonstrated trajectories. The state space of different options $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_N\}$ can be different, as long as there exists a direct or learnable mapping between $\mathcal{S}_i$ and $\mathcal{S}_j$ for some part of the space.

We aim to answer the question whether we can construct a hybrid hierarchical policy $\pi_\Omega(\omega_t|s_t)$ that can plan the next option o_{ω_t} that needs to be executed to bring the current state s_t to some desired s_{final} by using the forward dynamics model $\mathcal{D}_\omega$ in an n-step MPC look-ahead using a goal metric $\mathcal{G}_K$ that evaluates how close s_{t+n} is to s_{final}.

4 Method

At a particular point s_t when o_ω is active, we can compute how successful is following the policy given these conditions up to a particular time horizon. The action given by the policy is $a_t = \pi_\omega(s_t)$, and following the dynamics model we can write that $s_{t+1} = \mathcal{D}_\omega(s_t, a_t) = \mathcal{D}_\omega(s_t, \pi_\omega(s_t))$. As the dynamics model is conditioned on the policy, we can simplify the notation to $s_{t+1} = \mathcal{D}_\omega(s_t)$. Chaining it for n steps in the future we obtain $s_{t+n} = \mathcal{D}_\omega \circ \mathcal{D}_\omega \circ \cdots \circ \mathcal{D}_\omega(s_t) = \mathcal{D}_\omega^n(s_t)$. Thus, a policy over policies can sequentially optimize

$$\pi_\Omega(\omega_t|s_t) = \arg\max_\omega \left(\mathbb{E}\left[\mathbf{1}_{\mathcal{I}_\omega}(s_t) \cdot \mathcal{G} \circ \mathcal{D}_\omega^n(s_t)\right]\right) \tag{1}$$

After choosing and evaluating the optimal π_Ω with respect the above criterion, another controller can be selected until the goal is reached.

5 Experimental Setup

We perform two sets of experiments to showcase the capability of using the structured hierarchical policy by performing MPC future predictions at each step on a simulated MDP problem and on a gear assembly task on the PR2 robot.

Simulated MDP In the first we use the standard 19-state random walk task as defined in [15] and seen on Figure. 2(a). The goal of the agent is to reach past the 19^{th} state and obtain the $+1$ reward. The action space of the agent is to go "left" or "right". There also exist 5 options defined as in Section. 3, with the following policies: (1-3) policies that go "right" with a different termination probabilities $\beta = \{0.9, 0.5, 0.2\}$; (4) random action; (5) policy with action to go "'left" with $\beta = 0.5$. We assume that there exists a noisy dynamics model $\mathcal{D}_\omega$ and the goal evaluation model $\mathcal{G}_{MDP}$, obtained from demonstrations, that have probability of mispredicting the current state or its value of 0.2.

Gear Assembly In this task the PR2 robot needs to assemble a part of the Siemens Challenge[1], which involves grasping a compound gear from a table, and placing it on a peg module held in the other hand of the robot. A human operator

[1]The challenge can be seen https://new.siemens.com/us/en/company/fairs-events/robot-learning.html

can come in proximity to the robot, interfering with the policy plan. We have received expert demonstrations of the task being performed, as well as access to a set of option that **(1)** picks the gear from the table; **(2)** quickly moves the left PR2 arm in proximity to the other arm; **(3)** cautiously navigates the left PR2 arm to the other avoiding proximity with humans; **(4)** inserts the gear on the peg module. Policy (2) relies exclusively on path planning techniques, (4) is fully neural networks learned and (1, 3) are a mixture between a neural recognition module recognizing termination criteria and motion planning for the policy. The options share a common state space of the robots' joint angles. The initiation set of all policies is $\mathcal{R}^{12}$. The terminal criteria β for $o_{1,2}$ is inversely proportional to the closeness to a human to the robot; for o_{2-4} is the proximity to a desired offset from the other robot hand.

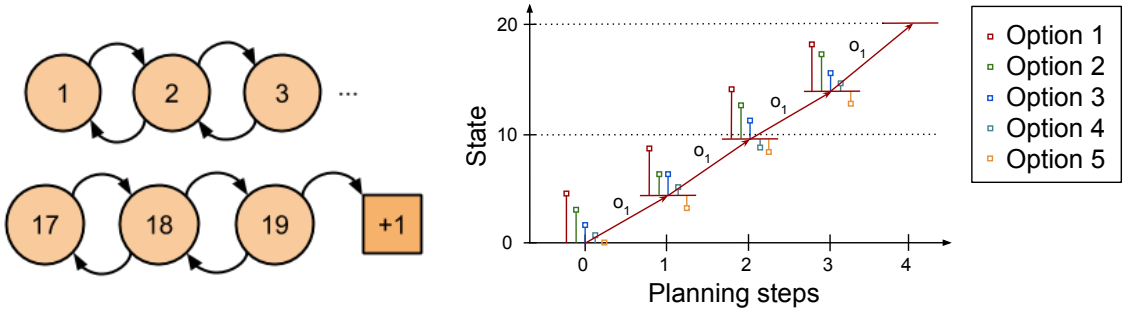
The dynamics model for each option is independent and is represented either as part of the motion planner, or similarly to the goal estimator - a neural networks working on the joint angle states of both arms of the robot.

In cases of options with overlapping initiation sets (i.e. options 1, 2, 3 all work within $\mathcal{R}^{12}$), we can softly partition the space of expected operation by fitting a Gaussian Mixture Model $\mathcal{F}_M$ on the trajectories of the demonstrations, where $s, s \sim \mathcal{F}_M$ is a sample state from the trajectory. $\mathcal{F}$ is a set of M Gaussian Mixtures $\mathcal{F} = \{N_i(\mu_i, \Sigma_i)|_{i=1..M}\}$, and J_k is a subset of $\mathcal{F}_M$, where samples from J_k correspond to samples from trajectory of option $k, 1 \leq k \leq N$. We can thus assess the likelihood of a particular option working in a state $s \sim \pi_j$ by evaluating $\mathcal{L}(s|\pi_j, \mathcal{F}_M) = \max_p [p(s|\mu_i, \Sigma_i)]_{\mu_i, \Sigma_i \in J}$ (5), This gives us the safety region, in which we expect the policy to work. By using the overlap between these regions, we can move the state of the system in a way that reaches the desired demonstrated configuration.

6 Results

We aim to demonstrate the viability of using the options dynamics as a method for choosing a satisfactory policy. The dynamics can be learned independently of the task, and can be used to solve a downstream task.

Simulated MDP The target solution shows the feasibility and compares the possible solutions by using different options. In Figure. 2(b), we can see that we reach the optimal state in just 4 planning steps, where each planning step is a rollout of an option. We can see the predicted state under the specified time horizon using different options. This naturally suggests the use of the policy π_1 that outperforms the alternatives (π_1 reaches state 6, π_2 - state 4, π_2 - state 3, π_3 - state 1, π_4 - state 1, π_5 - state 0). Even though the predicted state differs from the true rollout of the policy, it allows the hierarchical controller to use the one, which would progress the state the furthest. The execution of some options (i.e. option 5 in planning steps 1, 2, 3) reverts the state of the world to a less desirable one. By using the forward dynamics, we can avoid sampling these undesirable options.

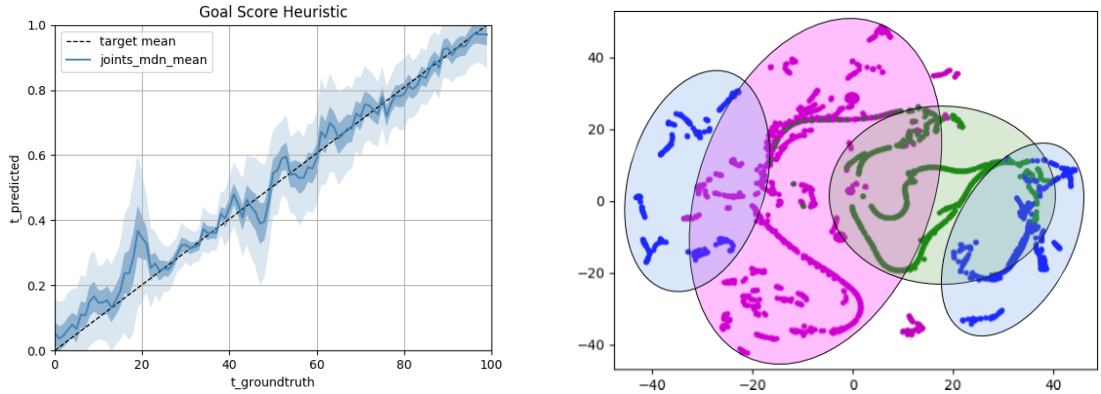


(a) MDP Problem (b) MDP Solution

Figure 2: (a) The 19-state MDP problem. The action space of the MDP is to move "left" or "right". The goal of the MDP problem is to reach past state 19 and obtain the +1 reward, which is equivalent to a termination state 20. (b) MDP solution. At timestep 0, a rollour of the 5 options is performed with the dynamics model. The expected resulting state is marked as blue vertical bars. The best performing option is used within the environment to obtain the next state - the red line at state 5 and planning step 1. This process is iterated until a desired state is reached.

Gear Assembly We obtained 10 demonstrations of the task being performed. In Figure. 3(a), we show the performance of the goal estimator network on an independent trial. We can observe that the state goal metric estimator closely tracks the expected ground truth values along the trajectory. This provides reasonable feedback that can be used by π_Ω to choose an appropriate next policy.

Similarly, in Figure. 3(b) we show the t-SNE of the trials of the robot trajectories that have no interruptions and some in which a human enters the scene and interferes with the motion of robot, forcing a change of policy to occur. We see that there is natural split in the states in which different options have been activated. We can notice that the overlap of the region of activation for the different policies allows the robot to grasp, navigate to, and insert the gear into the assembly by following these basins of the policies initiation. By following Eq.5 we can therefore create state space envelope of action of each option. The corresponding part of the state space, conditioned on the executed option, can have the safety constraints enforced by the underlying control method for the option.



(a) Goal Heuristic (b) t-SNE visuazlization of the controllers states.

Figure 3: (a) The learned heuristics about how close the current state is to the demonstrated goal state. (b) t-SNE plot of the controllers state during a set of trajectories. *Magenta* - o_1 for grasping the object, *Green* - o_2 and o_3 for navigating to the assembly with and without a human intervention and *Blue* - o_4 for inserting the gear onto the peg. The shaded regions illustrate the regions of control for the different policies.

7 Conclusion

We present DynoPlan - a hybrid hierarchical controller where by extending the options framework, we can rephrase the learning of a top level controller to an MPC planning solution. By unrolling the future states of each option, where each can be assessed on the contribution of furthering the agents intent based on the goal heuristic, we can choose the one best satisfying the problem requirements. This method of action selection allows to combine motion planning with neural network control policies in a single system, whilst retaining the completeness and performance guarantees of the work space of the associated options.

References

[1] Garrett Thomas, Melissa Chien, Aviv Tamar, Juan Aparicio Ojea, and Pieter Abbeel. Learning robotic assembly from cad. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[2] Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. Learnings options end-to-end for continuous action tasks. *arXiv preprint arXiv:1712.00004*, 2017.

[3] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009.

[4] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[5] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[6] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.

[7] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[8] Doina Precup. Eligibility traces for off-policy policy evaluation. *CS Department Faculty Publication Series*, 2000.

[9] Glenn A Iba. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3(4):285–317, 1989.

[10] P. Rumschinski S. Streif R. Findeisen P. Andonov, A. Savchenko. Controller verification and parametrization subject to quantitative and qualitative requirements. *IFAC-PapersOnLine*, 48(8):1174 – 1179, 2015.

[11] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] Jessie Huang, Fa Wu, Doina Precup, and Yang Cai. Learning safe policies with expert guidance. *arXiv preprint arXiv:1805.08313*, 2018.

[13] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[14] Daniel S Brown, Yuchen Cui, and Scott Niekum. Risk-aware active inverse reinforcement learning. *arXiv preprint arXiv:1901.02161*, 2019.

[15] Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Nowé. Learning with options that terminate off-policy. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Performance metrics for a physically-situated stimulus response task

Paul B. Reverdy*
Department of Aerospace and Mechanical Engineering
University of Arizona
Tucson, AZ 85721
preverdy@mail.arizona.edu

Abstract

Motivated by reactive sensor-based motion control problems that are ubiquitous in robotics, we consider a physically-situated stimulus response task. The task is analogous to the moving dots task commonly studied in humans, where subjects are required to determine the sign of a noisy stimulus and respond accordingly. In our physically-situated task, the robot responds by navigating to one of two predetermined goal locations. Our task is carefully designed to decouple the robot's sensory inputs from its physical dynamics. This decoupling greatly facilitates performance analysis of control strategies designed to perform the task.

We develop two strategies for performing the task: one that attempts to anticipate the correct action and another that does not. We consider two metrics of task performance; namely, total time required for the robot to reach a goal location and the total distance traveled in doing so. We derive semi-analytical expressions for the expected values of the two performance metrics. Using these expressions, we show that the anticipatory strategy reaches the goal location more quickly but results in the robot traveling a greater average distance, which corresponds to exerting greater physical effort. We suggest that this tradeoff between reaction time and physical effort is a fundamental tension in physically-situated stimulus-response tasks.

Keywords: Perceptual decision making, drift-diffusion model, affordance competition, robotics

Acknowledgements

This work has been supported by US Air Force Research Laboratory grant FA8650-15-D-1845 subcontract 669737-6.

*Web: https://www.paulreverdy.com

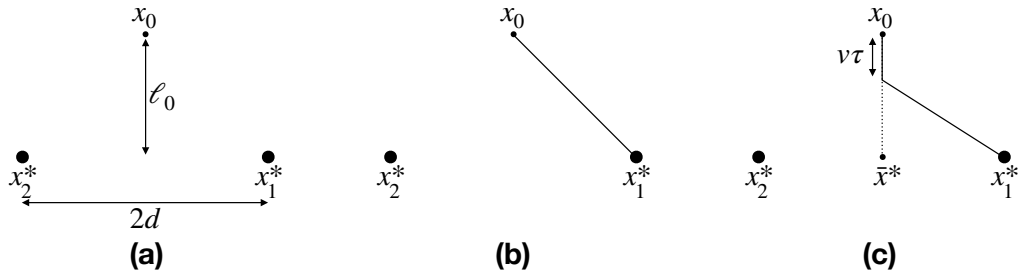


Figure 1: Task geometry. (a) The robot is initially located at $x_0 \in \mathbb{R}^2$ and needs to travel to either x_1^* or x_2^* depending on a noisy signal. (b) In strategy 1, the robot only moves after deciding and follows the trajectory shown. (c) In strategy 2, the robot moves towards $\bar{x}^*$ at speed v until deciding at time τ, then travels to the chosen goal.

1 Introduction

Robots often need to choose an appropriate behavior in response to some stimulus from the environment. Sensors are often mounted on the robot itself, so the information gain from sensing depends on the physical state of the robot. This coupling greatly complicates analysis of the control problem. Here we consider a task with simple sensing that decouples the sensor from the physical state.

Stimulus response tasks have been extensively studied in the psychology and neuroscience literature. A standard model with optimality properties is the drift-diffusion model (DDM) [1]. In the psychology literature, studies generally focus the decision-making process and assume that the action of reporting the decision requires some constant response time [1, 2]. In contrast, physically-situated tasks require costly movement and different decisions require different physical actions. The so-called affordance competition hypothesis [3], [4] suggests one mechanism for negotiating decisions in such physical contexts. In previous work [5], we began to develop a control architecture termed *motivation dynamics* for implementing a form of affordance-competition-like decision making in robot systems. In subsequent work [6], we began to investigate applying the motivation dynamics control architecture to reactive sensor-based motion planning.

2 A physically-situated stimulus response task

Key to developing metrics for physically-situated stimulus response tasks is choosing a task whose analysis is tractable. We consider a scenario where the robot is a point x in the Euclidean plane $\mathbb{R}^2$. The robot is equipped with a sensor which measures an environmental signal $e(t)$:

$$e(t) = s(t) + \sigma z(t), \tag{1}$$

with $s(t) \in \mathbb{R}, \sigma > 0$, and $z(t)$ iid Gaussian noise. Thus, $e(t) \sim \mathcal{N}(s(t), \sigma)$.

The robot is to perform a stimulus response task. In particular, if $s(t) = \mu > 0$, the robot is to drive to position $x = x_1^*$, while if $s(t) = -\mu < 0$, it should drive to $x = x_2^*$. Essentially, the robot is to carry out a sign test on its observed signal data and respond according to the perceived sign of the true stimulus $s(t)$.

We assume that the response locations x_1^*, x_2^* are separated by a distance $2d$ and that the robot's initial physical state $x(0) = x_0$ is a distance ℓ_0 away from the midpoint $\bar{x}^* = (x_1^* + x_2^*)/2$ in the transverse direction, as shown in Figure 1. Note that we have chosen this symmetric initial condition merely to simplify analysis and presentation.

3 Decision-making apparatus

The optimal decision-making scheme is the DDM, i.e., the continuum limit of the SPRT. Let $y \in \mathbb{R}$ be the accumulator state of the DDM. Then y obeys the SDE

$$\mathrm{d}y = A\mathrm{d}t + c\mathrm{d}W, \;\; y(0) = y_0 = 0. \tag{2}$$

Decisions are made when the state y crosses one of two thresholds $\pm Z \in \mathbb{R}$. Crossing $+Z > 0$ means go to location 1; crossing $-Z < 0$ implies go to location 2. Let τ represent the time at which the decision is made.

Let $p_+(\tau)$ be the distribution of $DT = \tau$ assuming that the positive boundary is associated with the correct response. It is well known that this distribution can be derived by solving the appropriate backward Kolmogorov or Fokker-Planck equation. The solution is usually expressed as a power series [2]:

$$p_+(\tau) = \frac{\pi}{4\alpha^2\beta} \exp\left(\alpha\beta - \beta\tau/2\right) \times \sum_{k=1}^{\infty} k \exp\left(-\frac{k^2\pi^2 t}{8\alpha^2\beta}\right) \sin\left(\frac{k\pi}{2}\right), \tag{3}$$

where $\alpha = Z/A$ is the scaled threshold and $\beta = (A/c)^2$ is the signal-to-noise ratio.

Denote the probability of crossing the correct, positive boundary by $1 - ER$, where

$$ER = \frac{1}{1 + \exp(2\alpha\beta)} \Rightarrow 1 - ER = \frac{\exp(2\alpha\beta)}{1 + \exp(2\alpha\beta)} \tag{4}$$

Analogously, let $p_-(\tau)$ be the distribution of $DT = \tau$ assuming that the accumulator crosses the negative (incorrect) threshold and produces an erroneous response. The two distributions p_+ and p_- can be related using ER [2]:

$$p_-(\tau) = \left(\frac{ER}{1 - ER}\right) p_+(\tau). \tag{5}$$

Note that both distributions p_+ and p_- have the same mean $\langle DT \rangle = \alpha \tanh(\alpha\beta)$ [2].

4 Control strategies

For simplicity in this initial work, we study two explicit strategies: **integrate-until-move** and **integrate-while-hedging**. Whenever the robot chooses to move, it does so at constant speed v. In the first strategy, called *integrate-until-move*, the robot performs the task by waiting to move until it has reached a decision. The robot decides by integrating stimulus information according to the DDM (2) until the accumulator state y crosses a threshold $\pm Z$. Upon reaching a decision, the robot travels directly to the corresponding goal state: if y crosses through the threshold $+Z$, the robot travels to goal 1 located at x_1^*, while if y crosses through the threshold $-Z$, the robot travels to goal 2 located at x_2^*. See Figure 1(b).

Note that this integrate-until-move control strategy results in the robot traveling the minimal distance required to complete the task. However, the time spent integrating stimulus information is wasted in the sense that the robot does not move until it has reached its desired level of certainty in its final decision. If the signal-to-noise ratio is low, the decision time may be significant relative to the the travel time required to carry out the physical action associated with the decision. In such a scenario, it is likely beneficial to begin moving before a certain decision can be reached. This motivates the second strategy presented below.

The second strategy we consider, called *integrate-while-hedging*, seeks to capture the benefit of moving before reaching a decision in a way that is analytically tractable. Given the geometry of the task shown in Figure 1(a), it is clear that one can anticipate moving towards either goal location by moving toward the point $\bar{x}^* = (x_1^* + x_2^*)/2$ that is the midpoint between the two goals. Therefore, moving towards $\bar{x}^*$ is a natural way to move while hedging, i.e., without committing to either decision.

This hedging observation motivates the following integrate-while-hedging control strategy. In this strategy, the robot performs the hedging action, i.e., moves towards $\bar{x}^*$, while integrating stimulus information. When the accumulator variable y crosses a threshold, the robot makes a decision and travels to the corresponding goal. If the robot reaches the midpoint x^* before making a decision, it stops moving until a decision can be reached. See Figure 1(c).

Note that, compared to the integrate-until-move strategy, the integrate-while-hedging strategy is likely to result in the robot traveling a larger distance. The advantage is that, by moving in such a way as to approach both goals while gathering information, the integrate-while-hedging strategy may result in a shorter overall response time. Quantifying this tradeoff between total time and travel distance is the subject of the next section.

5 Performance metrics

We analyze performance in terms of the two quantities: total distance traveled and total response time. Note that the total response time required to perform the task is the sum of decision time and travel time.

For the purposes of analysis, assume that the initial physical state at time $t = 0$ is as given in Figure 1(a) and that the initial accumulator state (cf. (2)) $y_0 = 0$. The stimulus $e(t)$ is assumed to be zero for $t < 0$. At time $t = 0$, the stimulus $s(t)$ takes value μ and remains constant for the duration of the task. Without loss of generality, we assume that $\mu > 0$, so the correct response is to travel to goal 1 located at x_1^*. Finally, we assume that the robot travels at a constant speed v whenever it decides to move.

We consider the task to end either a) when the accumulator state crosses the negative threshold and the robot makes an incorrect decision, or b) when the robot reaches the correct goal state x_1^*. Since the decision-making process is independent of the robot's physical state x, the probability of the task ending due to an incorrect decision is given by the error rate expression (4) for either of the two control strategies. Note that the error rate can be controlled by selecting the threshold Z. Therefore, we focus on distance traveled and total response time conditional on the robot selecting the correct response.

5.1 Travel distance

We first consider the total distance traveled. Let τ be the random variable denoting the first passage (i.e., decision) time for the DDM decision process (2). When using the integrate-then-move control strategy, the robot's motion is the same for any correct decision, and will result in traveling a total distance

$$D_1 = \sqrt{d^2 + \ell_0^2}. \tag{6}$$

Conditional on making the correct decision, the quantity D_1 is deterministic.

When using the integrate-while-hedging strategy, the robot first travels towards $\bar{x}^*$ at speed v until it makes a decision at time τ. The distance traveled before making a decision is given by $\min(v\tau, \ell_0)$, where the minimum operation results from the robot stopping at $\bar{x}^*$ if it reaches that point before making a decision. After deciding, the robot moves directly to x_1^*, which requires traveling a distance

$$\sqrt{(\ell_0 - \min(v\tau, \ell_0))^2 + d^2} = \sqrt{\max(\ell_0 - v\tau, 0)^2 + d^2}.$$

The total distance traveled in this scenario is

$$D_2 = \min(v\tau, \ell_0) + \sqrt{\max(\ell_0 - v\tau, 0)^2 + d^2}. \tag{7}$$

Note that the quantity D_2 is a random variable because it depends on the random decision time τ.

5.2 Total time

Now, consider the total time required for the response using the two strategies. Recall that τ is the random variable denoting the first passage time of the decision process (2). When using the integrate-then-move strategy, the travel time is D_1/v, which is deterministic conditional on making the correct decision. Thus, the total response time using the integrate-then-move strategy is

$$T_1 = \tau + D_1/v = \tau + \sqrt{d^2 + \ell_0^2}/v, \tag{8}$$

where the only source of randomness is the decision time τ that appears additively.

When using the integrate-while-hedging strategy, the travel time after making a decision is $\sqrt{\max(\ell_0 - v\tau, 0)^2 + d^2}/v$. The total response time is

$$T_2 = \tau + \sqrt{\max(\ell_0 - v\tau, 0)^2 + d^2}/v, \tag{9}$$

where the randomness from τ enters in a nonlinear way.

5.3 Expected performance

The total distance and total time metrics evaluated in (7), (8), and (9) are random variables due to their dependence on the random decision time τ. To facilitate comparison between the two strategies, we consider the expected values of the various metrics. Consider first the integrate-then-move strategy. Let ED_1 and ET_1 be the expected values of D_1 and T_1. We have

$$ED_1 = D_1 = \sqrt{d^2 + \ell_0^2} \tag{10}$$

since D_1 is deterministic. As noted above in (5), the expected value of τ is equal to $\alpha \tanh(\alpha\beta)$ and this expected value is also equal to the conditional expected value of τ given that the positive (correct) threshold is the one that is crossed. Thus, the expected value of T_1 is given by

$$ET_1 = \alpha \tanh(\alpha\beta) + \sqrt{d^2 + \ell_0^2}/v. \tag{11}$$

Now, consider the integrate-while-hedging strategy. Let ED_2 and ET_2 be the expected values of D_2 and T_2. The nonlinear way in which τ enters in the expressions for D_2 and T_2 means that computing their expected values requires evaluating integrals that do not appear to have analytical solutions. Thus, we resort to numerical approximations. In particular, we compute the quantities

$$ED_2 = \mathbb{E}\left[D_2 | \text{correct response}\right] = \frac{1}{1 - ER} \int_0^\infty D_2 p_+(\tau) \mathrm{d}\tau, \text{ and} \tag{12}$$

$$ET_2 = \mathbb{E}\left[T_2 | \text{correct response}\right] = \frac{1}{1 - ER} \int_0^\infty T_2 p_+(\tau) \mathrm{d}\tau, \tag{13}$$

where ER is the error rate (4) required to condition on the robot selecting the correct response and $p_+(\tau)$ is given by (3).

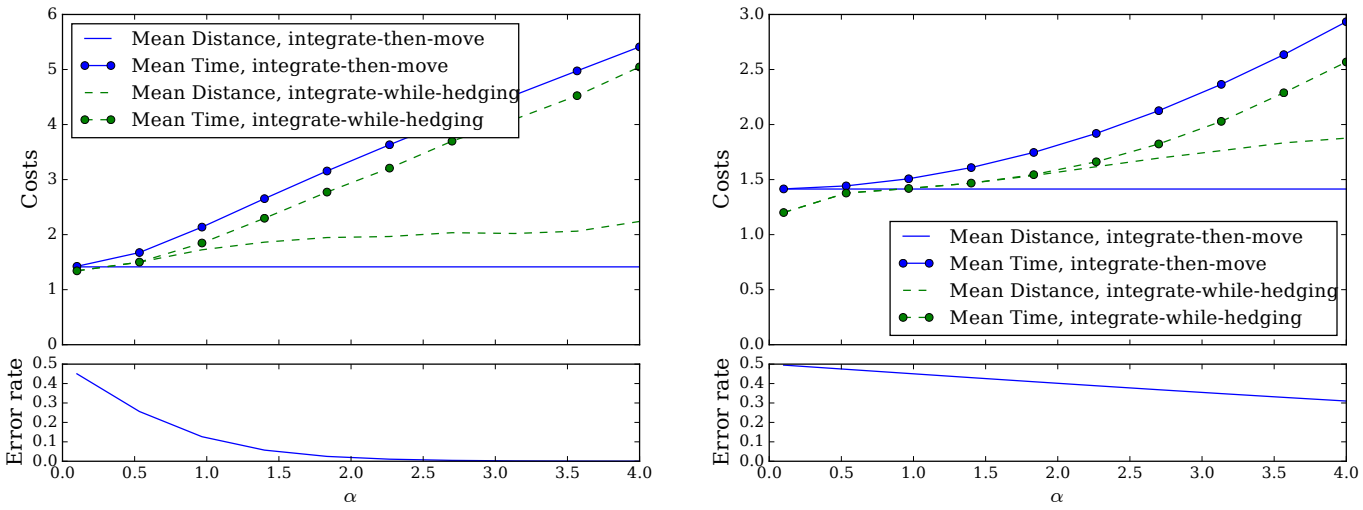


Figure 2: Error rate (4); expected total distance (10), (12); and expected total time (11), (13) for the two strategies as a function of normalized threshold α. Panel (a) shows data from a high signal-to-noise ratio environment with $\beta = (A/c)^2 = 1$; (b) a low signal-to-noise ratio environment with $\beta = 0.1$. Intuitively, the integrate-while-hedging strategy results in smaller expected total time (i.e., a faster response) at the cost of a larger expected total distance traveled. The other problem parameters were set to $d = \ell_0 = 1$ and $v = 1$, respectively.

Figure 2 shows the results of numerically evaluating the error rate (4) and cost metrics (10)–(13) in two different environments. In both panels, the physical problem parameters were set to $d = \ell_0 = 1$ and $v = 1$, respectively, and the various metrics are plotted as functions of the normalized threshold $\alpha = Z/A$. Panel (a) shows results for an environment with high signal-to-noise ratio, $\beta = 1$, while panel (b) shows results for an environment with low signal-to-noise ratio, $\beta = 0.1$. Intuitively, the integrate-while-hedging strategy results in smaller expected total time (i.e., a faster response) at the cost of a larger expected total distance traveled.

In future work, we intend to study a more complete set of control strategies for this task in order to find a set of optimal strategies. It is likely that the set of optimal strategies trade off between the two metrics we consider, much as statistical hypothesis tests trade off between the probability of type I and type II errors. A single optimal strategy can then be found by selecting an optimal balance between these two metrics, e.g. in terms of relative costs of movement versus idle time. We postulate that the motivation dynamics control framework studied in [5, 6] will outperform both the integrate-then-move and the integrate-while-hedging strategies.

References

[1] R. Bogacz, E. Brown, J. Moehlis, P. Holmes, and J. D. Cohen, "The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks." *Psychological review*, vol. 113, no. 4, p. 700, 2006.

[2] K. F. Wong-Lin, P. Holmes, and T. Broderick, "Closed-Form Approximations of First-Passage Distributions for a Stochastic Decision-Making Model," *Applied Mathematics Research eXpress*, vol. 2009, no. 2, pp. 123–141, 02 2010. [Online]. Available: https://dx.doi.org/10.1093/amrx/abp008

[3] P. Cisek and J. F. Kalaska, "Neural mechanisms for interacting with a world full of action choices," *Annual review of neuroscience*, vol. 33, pp. 269–298, 2010.

[4] N. F. Lepora and G. Pezzulo, "Embodied choice: how action influences perceptual decision making," *PLoS computational biology*, vol. 11, no. 4, p. e1004110, 2015.

[5] P. B. Reverdy and D. E. Koditschek, "A dynamical system for prioritizing and coordinating motivations," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 2, pp. 1683–1715, 2018.

[6] P. B. Reverdy, V. Vasilopoulos, and D. E. Koditschek, "Motivation dynamics for autonomous composition of navigation tasks," *In preparation*, 2019.

Belief space model predictive control for approximately optimal system identification

Boris Belousov
Department of Computer Science
Technische Universität Darmstadt, Germany
belousov@ias.tu-darmstadt.de

Hany Abdulsamad
Department of Computer Science
Technische Universität Darmstadt, Germany
abdulsamad@ias.tu-darmstadt.de

Matthias Schultheis
Department of Computer Science
Technische Universität Darmstadt, Germany
matthias.schultheis@gmail.com

Jan Peters
Department of Computer Science
Technische Universität Darmstadt, Germany
Max Planck Institute for Intelligent Systems
peters@ias.tu-darmstadt.de

Abstract

The fundamental problem of reinforcement learning is to control a dynamical system whose properties are not fully known in advance. Many articles nowadays are addressing the issue of optimal exploration in this setting by investigating the ideas such as curiosity, intrinsic motivation, empowerment, and others. Interestingly, closely related questions of optimal input design with the goal of producing the most informative system excitation have been studied in adjacent fields grounded in statistical decision theory. In most general terms, the problem faced by a curious reinforcement learning agent can be stated as a sequential Bayesian optimal experimental design problem. It is well known that finding an optimal feedback policy for this type of setting is extremely hard and analytically intractable even for linear systems due to the non-linearity of the Bayesian filtering step. Therefore, approximations are needed. We consider one type of approximation based on replacing the feedback policy by repeated trajectory optimization in the belief space. By reasoning about the future uncertainty over the internal world model, the agent can decide what actions to take at every moment given its current belief and expected outcomes of future actions. Such approach became computationally feasible relatively recently, thanks to advances in automatic differentiation. Being straightforward to implement, it can serve as a strong baseline for exploration algorithms in continuous robotic control tasks. Preliminary evaluations on a physical pendulum with unknown system parameters indicate that the proposed approach can infer the correct parameter values quickly and reliably, outperforming random excitation and naive sinusoidal excitation signals, and matching the performance of the best manually designed system identification controller based on the knowledge of the system dynamics.

Keywords: Bayesian experimental design, active exploration, curiosity, belief space planning, trajectory optimization

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640554.

1 Introduction and related work

Adaptation and learning arise as a by-product of optimization in the belief space within the framework of Bayesian decision theory [Stratonovich, 1968a, Stratonovich, 1968b]. In modern terminology, learning is planning in a partially observable Markov decision process [Asmuth and Littman, 2011]. We pursue this line of reasoning and frame the problem of pure exploration (i.e., without any extrinsic reward) as a problem of online belief space trajectory optimization.

Optimal system identification and experimental design [Mehra, 1974, Bombois et al., 2011, Ryan et al., 2016] pursue a similar objective. They seek an optimal exploration strategy in stochastic sequential decision making problems. Contrary to the generic solution based on approximate dynamic programming [Feldbaum, 1960, Huan and Marzouk, 2016], we do not aim to find an optimal parametric policy but instead let a belief space planner choose the most explorative actions.

Approaches to (approximately) optimal system identification based on model predictive control (MPC) have been studied before [Larsson et al., 2013]. Algorithmically, our method is most closely related to [Kahn et al., 2015], who also used direct transcription in the belief space for trajectory optimization. However, what is different in our case is the objective function and its particular decomposition into a sum of terms that facilitates computation. More concretely, since robot dynamics is linear in the physics parameters [Atkeson, 1989], we can perform Bayesian inference in closed form.

The paper is structured as follows: Section 2 introduces the approach, Section 3 provides evaluations, and Section 4 highlights future directions.

2 Belief space optimization for system identification

Consider a dynamical system of the following form

$$x' = Ax + B(x,u)\theta \tag{1}$$

where $x \in \mathbb{R}^n$ is the current state, $u \in \mathbb{R}^m$ is the current action, $x' \in \mathbb{R}^n$ is the next state, matrix $A \in \mathbb{R}^{n\times n}$ is constant and matrix $B(x,u) \in \mathbb{R}^{n\times m}$ is dependent on the state and action. Many classical continuous control environments can be written in this way.

2.1 Example: pendulum dynamics

As a concrete instantiation of (1), consider the dynamics of a pendulum

$$\ddot{q} = \phi\left(\begin{pmatrix} q \\ \dot{q} \end{pmatrix}, u\right)^T \theta = \begin{pmatrix} -\sin(q+\pi) & -\dot{q} & u \end{pmatrix} \begin{pmatrix} \frac{3g}{2l} \\ \frac{3b}{ml^2} \\ \frac{3}{ml^2} \end{pmatrix} \tag{2}$$

with mass m, length l, and gravity g. The state of the pendulum $x = (q, \dot{q})$ is comprised of the angle q and the angular velocity $\dot{q}$. Crucially, the kinematic parameters $\phi(x,u)$ and the dynamic parameters θ separate. The system can be discretized using the implicit Euler integration scheme

$$x' = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} x + \begin{pmatrix} h^2 \\ h \end{pmatrix} \phi(x,u)^T \theta. \tag{3}$$

This representation directly corresponds to the generic form (1), with matrices A and $B(x,u)$ straightforward to identify.

2.2 Propagation of uncertainty

If parameter values θ are uncertain, they should be characterized by a probability distribution $p(\theta)$. The full state of the system should then include it and we have to describe its dynamics. Assuming the initial belief $p(\theta) = N(\theta|\mu, \Sigma)$ and the system dynamics $p(x'|x,u;\theta) = N(x'|Ax + B(x,u)\theta, Q)$ are Gaussian, the posterior after observing a transition (x,u,x') is also Gaussian with parameters given by the standard Kalman filter update equations [Bishop, 2006]

$$K(x,u,\Sigma) = \Sigma B(x,u)^T \left(Q + B(x,u)\Sigma B(x,u)^T\right)^{-1}, \tag{4}$$

$$L(x,u,\Sigma) = I - K(x,u,\Sigma)B(x,u), \tag{5}$$

$$\mu' = \mu + K(x,u,\Sigma)\left(x' - Ax - B(x,u)\mu\right), \tag{6}$$

$$\Sigma' = L(x,u,\Sigma)\Sigma. \tag{7}$$

Kalman gain $K(x,u,\Sigma)$ and matrix $L(x,u,\Sigma)$ are introduced for convenience to simplify Equations (6) and (7) that describe the dynamics of the sufficient statistics of the belief state.

To plan using the model (6)-(7), future observations x' need to be integrated out. This results in the maximum likelihood transition dynamics $x' = Ax + B\mu$ and the constant mean update $\mu' = \mu$. Intuitively, such constancy is a manifestation of the fact that the mean of the parameter estimate μ cannot be improved before observing any data. Nevertheless, its variance Σ can be controlled.

Equation (7) gives the update rule for the covariance matrix and serves as the key to our formulation of the objective function. Namely, we exploit the fact that the covariance matrix at the next time step is given by a product of matrices. For example, after two time steps, $\Sigma'' = L(x', u', \Sigma')L(x, u, \Sigma)\Sigma$.

2.3 Entropy minimization objective

What should the objective function be? A conceptually straightforward approach is to minimize the entropy of the posterior distribution over the parameters at the end of the planning horizon. This objective essentially asks for the most informative actions and can be identified with the information gain criterion [Lindley et al., 1956]. It also fits nicely with the multiplicative form of the covariance matrix, turning the product into a sum. For example, for a two-stage problem,

$$J = \frac{1}{2}\log\det\left(2\pi e\Sigma''\right) \propto \log\det\Sigma'' = \log\det L(x', u', \Sigma') + \log\det L(x, u, \Sigma) + \log\det\Sigma. \tag{8}$$

Similarly, for an N-step trajectory,

$$J \propto \sum_{k=0}^{N-1} \log\det L(x_k, u_k, \Sigma_k). \tag{9}$$

Thus, the summand $L(x_k, u_k, \Sigma_k)$ can be viewed as a running cost. Adding a regularization term $u^T Ru$ for smoothness, we arrive at the following optimization problem

$$\begin{aligned}
\underset{u_{0:N-1}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \log\det L(x_k, u_k, \Sigma_k) + u_k^T R u_k && (10)\\
\text{subject to} \quad & x_{k+1} = Ax_k + B(x_k, u_k)\mu, \quad k = 0, 1, \ldots, N-1, && (11)\\
& \Sigma_{k+1} = L(x_k, u_k, \Sigma_k)\Sigma_k, \quad k = 0, 1, \ldots, N-1, && (12)
\end{aligned}$$

where $L(x, u, \Sigma) = I - \Sigma B(x, u)^T \left(Q + B(x, u)\Sigma B(x, u)^T\right)^{-1} B(x, u)$. This problem can be directly plugged into a trajectory optimizer, e.g., CasADi [Andersson et al., 2012]; state and control constraints can be added if needed.

3 Evaluation

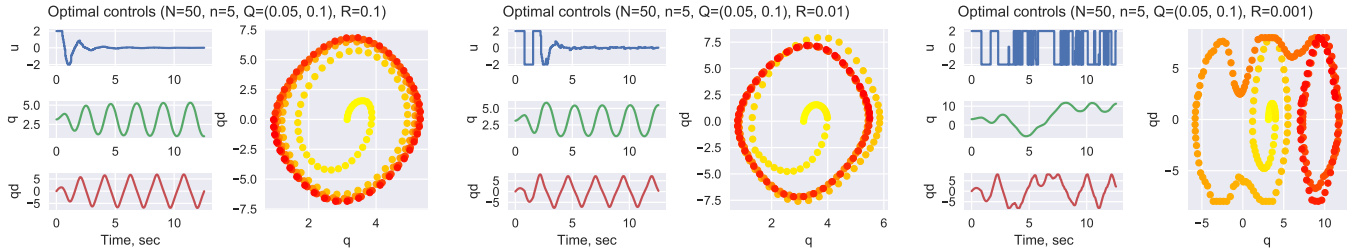
Having solved the problem above, we obtain a sequence of actions $u_{0:N-1}$ that should reveal the most about the system. Note that this sequence of actions depends on our prior belief $p(\theta|\mu, \Sigma)$ because μ enters the state dynamics and Σ figures in the covariance cost. Thus, the optimal sequence of actions is a function of the prior together with the initial state x_0, i.e., $u_{0:N-1} = \psi(x_0, \mu, \Sigma)$. We can think of ψ as a call to the trajectory optimizer.

The main question is whether this sequence of actions is better than any other one given that the true value $\mu^\star$ is different from μ. One way to evaluate this hypothesis is to execute $u_{0:N-1}$ on the real system with parameters $\mu^\star$ and then find the posterior $p(\theta|x_{0:N}, u_{0:N-1})$ given the observed trajectory. An even better solution is to replan after every time step. Such closed loop control should intuitively speed up convergence to the true parameter value. We call this approach belief space model predictive control for approximately optimal system identification.

We compare the belief space MPC approach (Figure 1) against random and sinusoidal excitations (Figure 2) on the pendulum environment from OpenAI Gym [Brockman et al., 2016]. Optimal exploration performs well and beats random actions and a naively chosen excitation signal by a large margin (Figure 3). However, a wisely chosen excitation signal can be as good as the optimal one (Figure 4). The optimization approach was found quite insensitive to the choice of the action cost R in a reasonable range, although extremely small values were found to cause instability.

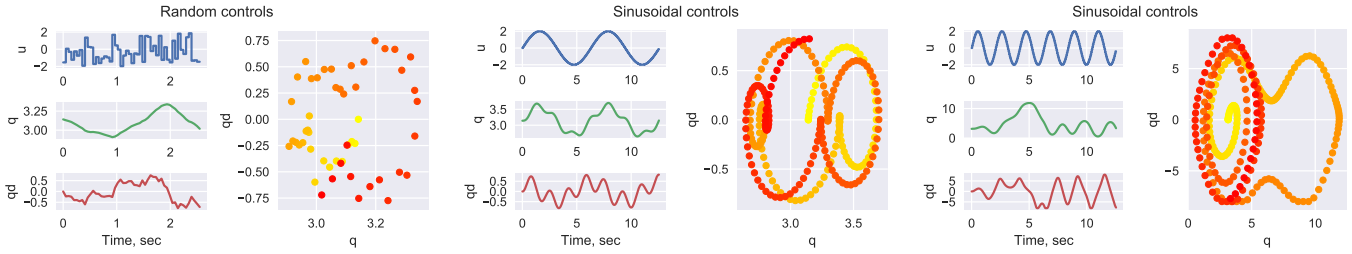
4 Conclusion

Although the preliminary results are encouraging, further investigation is required. First, evaluation on more complex systems must be performed to demonstrate the scalability of the approach. Second, comparison to other exploration strategies is needed to better understand the trade-offs between optimality and heuristics. Third, the assumption on the system dynamics (1) can be relaxed to allow for more flexible models; for example, the feature mapping ϕ can be learned by exploiting its invariance to dynamics parameters, or a non-parametric model, such as a Gaussian process, can be employed to represent the system dynamics.



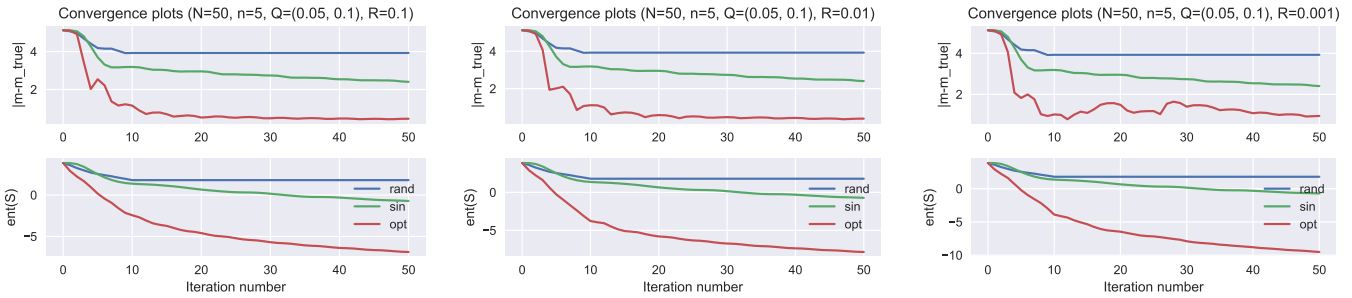
(a) high action cost, $R = 0.1$ (b) medium action cost, $R = 0.01$ (c) low action cost, $R = 0.001$

Figure 1: Effects of the action cost R on the closed-loop system performance. Trajectories are executed on Pendulum-v0 using belief-space MPC with horizon N and replanning every n steps. System noise Q is fixed and the action cost R is varying. Three scenarios are shown. In (a), the action cost is high, therefore the controller quickly pumps the energy into the system and fades away to observe the oscillations; this is possible because Pendulum-v0 is frictionless (although the controller has a non-zero prior on the friction coefficient). In (b), the cost of actions is lower, therefore the controller can enjoy taking larger actions a bit longer. In (c), the controller gets unstable, probably because the reward function is quite flat without action regularization and the action limits are too small to escape the flat region.



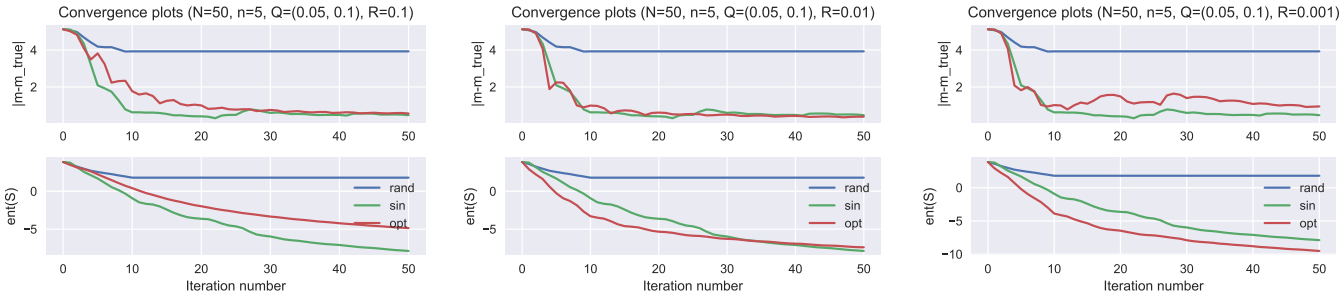
(a) random actions do not explore (b) slow sinusoid — sufficient excitation (c) fast sinusoid — best coverage

Figure 2: Compared to the optimal controls, random actions (a) perform very badly because they fail to explore the state space. On the other hand, a naive sinusoidal signal (b) works quite well on the pendulum, making it swing in all kinds of ways. However, the quality of system identification crucially depends on finding the right frequency of the sinusoid. A more oscillatory signal (c) turns out to be better for system identification (see convergence plots below).



(a) high action cost, $R = 0.1$ (b) medium action cost, $R = 0.01$ (c) low action cost, $R = 0.001$

Figure 3: Convergence plots show how quickly the posterior concentrates around the true parameter value; convergence in terms of distance from the mean and in terms of entropy of the posterior are shown. The posterior is updated after every n steps in the environment with the newly obtained data; one iteration on the x-axis corresponds to one posterior update. Three excitation signals are compared: random actions (blue), slow sinusoid (green), and optimal controls (red). Three scenarios are displayed from left to right that correspond to different action costs; only the red curve is different among the subplots, the other two curves are the same and kept for reference. All subplots demonstrate that the optimal excitation controls are significantly better than random or sinusoidal ones. Subplots (a) and (b) show similar red curves, which means that optimization is insensitive to the choice of the action cost in a reasonable range. Subplot (c) demonstrates that extremely low action costs may lead to oscillations; also observe that the final entropy in (c) is lower, meaning that the controller is more certain in the end.



(a) high action cost, $R = 0.1$ (b) medium action cost, $R = 0.01$ (c) low action cost, $R = 0.001$

Figure 4: A properly chosen excitation signal can yield very good results. These plots show that using a faster sinusoid (green), one can obtain as good parameter estimates as with an optimal signal. In (a), the fast sinusoid discovers the correct value faster and in the end it is even more certain than the optimal controller. In (b), both the optimal controls and the sinusoid perform on par. In (c), the posterior mean found with the optimal actions is further away from the true value and at the same time the controller is more confident about it; this shows the importance of the choice of costs.

References

[Andersson et al., 2012] Andersson, J., Åkesson, J., and Diehl, M. (2012). Casadi: A symbolic package for automatic differentiation and optimal control. In *Recent advances in algorithmic differentiation*, pages 297–307. Springer.

[Asmuth and Littman, 2011] Asmuth, J. and Littman, M. (2011). Learning is planning: near bayes-optimal reinforcement learning via monte-carlo tree search. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 19–26. AUAI Press.

[Atkeson, 1989] Atkeson, C. G. (1989). Learning arm kinematics and dynamics. *Annual review of neuroscience*, 12(1):157–183.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

[Bombois et al., 2011] Bombois, X., Gevers, M., Hildebrand, R., and Solari, G. (2011). Optimal experiment design for open and closed-loop system identification. *Communications in Information and Systems*, 11(3):197–224.

[Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[Feldbaum, 1960] Feldbaum, A. (1960). Dual control theory. i. *Avtomatika i Telemekhanika*, 21(9):1240–1249.

[Huan and Marzouk, 2016] Huan, X. and Marzouk, Y. M. (2016). Sequential bayesian optimal experimental design via approximate dynamic programming. *arXiv preprint arXiv:1604.08320*.

[Kahn et al., 2015] Kahn, G., Sujan, P., Patil, S., Bopardikar, S., Ryde, J., Goldberg, K., and Abbeel, P. (2015). Active exploration using trajectory optimization for robotic grasping in the presence of occlusions. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4783–4790. IEEE.

[Larsson et al., 2013] Larsson, C. A., Annergren, M., Hjalmarsson, H., Rojas, C. R., Bombois, X., Mesbah, A., and Modén, P. E. (2013). Model predictive control with integrated experiment design for output error systems. In *2013 European Control Conference (ECC)*, pages 3790–3795. IEEE.

[Lindley et al., 1956] Lindley, D. V. et al. (1956). On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005.

[Mehra, 1974] Mehra, R. (1974). Optimal input signals for parameter estimation in dynamic systems–survey and new results. *IEEE Transactions on Automatic Control*, 19(6):753–768.

[Ryan et al., 2016] Ryan, E. G., Drovandi, C. C., McGree, J. M., and Pettitt, A. N. (2016). A review of modern computational algorithms for bayesian optimal design. *International Statistical Review*, 84(1):128–154.

[Stratonovich, 1968a] Stratonovich, R. (1968a). *Conditional Markov processes and their application to the theory of optimal control*. Elsevier.

[Stratonovich, 1968b] Stratonovich, R. (1968b). Is there a theory of synthesis of optimal adaptive, self learning and self adjusting systems? *Avtomat. i Telemekh*, 29(1):96–107.

Momentum and mood in policy-gradient reinforcement learning

Daniel Bennett
Princeton Neuroscience Institute
Princeton University
Princeton, NJ 08544
daniel.bennett@princeton.edu

Guy Davidson
College of Computational Sciences
Minerva Schools at KGI
San Francisco, CA 94103
guy@minerva.kgi.edu

Yael Niv
Princeton Neuroscience Institute and Department of Psychology
Princeton University
Princeton, NJ 08544
yael@princeton.edu

Abstract

Policy-gradient reinforcement learning (RL) algorithms have recently been successfully applied in a number of domains. In spite of this success, however, relatively little work has explored the implications of policy-gradient RL as a model of human learning and decision making. In this project, we derive two new policy-gradient algorithms that have implications as models of human behaviour: TD(λ) Actor-Critic with Momentum, and TD(λ) Actor-Critic with Mood. For the first algorithm, we review the concept of momentum in stochastic optimization theory, and show that it can be readily implemented in a policy-gradient RL setting. This is useful because momentum can accelerate policy gradient RL by filtering out high-frequency noise in parameter updates, and may also confer a degree of robustness against convergence to local maxima in reward. For the second algorithm, we show that a policy-gradient RL agent can implement an approximation to momentum in part by maintaining a representation of its own mood. As a proof of concept, we show that both of these new algorithms outperform a simpler algorithm that has neither momentum nor mood in a standard RL testbed, the 10-armed bandit problem. We discuss the implications of the mood algorithm as a model of the feedback between mood and learning in human decision making.

Keywords: actor-critic, policy gradient, momentum, mood

1 Introduction

RL algorithms can be divided into three broad classes: value-based, policy-based, and actor-critic. Algorithms in these classes are distinguished by whether they learn just a value function, and then use this value function to generate a policy (value-based algorithms), learn only a policy function and no value function (policy-based algorithms), or learn both a value function and an independent policy function (actor-critic algorithms). In psychology and neuroscience, recent studies of learning and decision making have tended to focus on the processes by which humans and other animals learn a value function. By contrast, comparatively little work has explored the implications for human learning and decision making of the different methods of policy improvement employed by policy-based and actor-critic algorithms, despite some evidence that human learning is consistent with policy updating rather than value updating [1].

Policy-gradient RL is a well-studied family of policy improvement methods that uses feedback from the environment to estimate the gradient of reinforcement with respect to the parameters of a differentiable policy function [2, 3]. This gradient is then used to adjust the parameters of the policy in the direction of increasing reinforcement. In recent years, RL algorithms that incorporate a policy-gradient component have been successfully applied to real-time robot control in a continuous action space [4], to Atari computer games [5], and to the board game Go [6].

Because it uses the gradient of reinforcement to update a policy, policy-gradient RL can be thought of as a form of stochastic gradient descent (SGD). As a result, theoretical advances in stochastic optimization theory concerning SGD methods can be applied directly to policy-gradient RL algorithms. For instance, it is known that SGD convergence can be greatly accelerated by a *momentum* term [7, 8, 9]. Here, we show that adding a momentum term to policy-gradient RL improves performance in a standard testbed, the 10-armed bandit problem, and we explore the implications of momentum as a model of phenomena in human learning and decision making. Specifically, we show that a mood variable—defined as an exponential moving average of reward prediction errors [10]—can be used to help approximate a momentum update without ever computing a momentum term explicitly.

2 Theoretical framework and background

2.1 Gradient descent and momentum

Gradient descent tries to find the parameters θ that minimize an objective function $J(\theta)$ by incrementally updating θ in the direction of $\nabla_\theta J(\theta)$, the gradient of J with respect to θ. The step size is controlled by a learning rate η:

$$\begin{aligned} u_t &= \eta \nabla_\theta J(\theta) \\ \theta_{t+1} &= \theta_t - u_t \end{aligned} \tag{1}$$

In many cases it may be computationally infeasible to calculate $\nabla_\theta J(\theta)$. *Stochastic* gradient descent therefore uses an estimated gradient $\nabla_\theta \widetilde{J}(\theta)$, typically calculated as the mean of a mini-batch of training samples. In practice, a *momentum* term [7, 8] is often also used to help overcome two distinct but related limitations of SGD. These limitations are, first, that SGD considers only the the slope of the objective function (first derivative) and not its curvature (second derivative). This can cause difficulty in the presence of 'ravines' in the objective function where the curvature of J differs with respect to different dimensions of θ [11]. Second, SGD's use of an estimated rather than an exact gradient increases the variance of parameter updates: although $\nabla_\theta \widetilde{J}(\theta)$ is equal to $\nabla_\theta J(\theta)$ in expectation, at any single time-step unsystematic error in gradient estimation can lead to suboptimal parameter updates.

Momentum addresses both of these limitations by updating parameters according to both the estimated gradient of J at the current θ and a proportion m of the parameter update at the previous timestep: $u_t = \eta \nabla_\theta \widetilde{J}(\theta) + m u_{t-1}$. In this way, momentum effectively implements a low-pass filter on parameter updates. This filtering resolves the limitations of SGD described above, because both oscillations resulting from differential curvature of J and unsystematic error in gradient estimation slow SGD by introducing high-frequency noise to parameter updates. For this reason, momentum has long been used in machine learning, especially in training neural networks by backpropagation [9].

2.2 Policy-gradient reinforcement learning

A policy-gradient RL algorithm performs gradient *ascent* on an objective function that evaluates its policy under current parameters (e.g., the expected future reward under the current policy). This is achieved by updating the parameters θ of a differentiable policy π_θ [2, 3] in the direction of the gradient of the objective function with respect to θ.

In this project we sought to explore the consequences of adding a momentum term to a policy-gradient RL algorithm. Our starting point was *TD(λ) Actor-Critic*, which implements policy-gradient RL via a form of advantage updating [4, 5]:

TD(λ) Actor-Critic. The critic learns a state-value function V using a learning rate α, and provides a scalar reward prediction error δ to the actor at each timestep. In turn, the actor uses this reward prediction error to improve its policy by updating θ in the direction of the product of δ and the accumulating eligibility trace e_t.

$\delta_t = r + V_{t-1}(s') - V_{t-1}(s)$	Calculate reward prediction error
$V_t(s) = V_{t-1}(s) + \alpha\delta_t$	Update state value
$e_t = \lambda e_{t-1} + \nabla_\theta \log_{\pi_\theta}(s, a)$	Increment eligibility trace
$u_t = \eta e_t \delta_t$	Calculate parameter update
$\theta_{t+1} = \theta_t + u_t$	Update parameters

TD(λ) Actor-Critic makes use of a variable called the score function: $\nabla_\theta \log \pi_\theta(s, a)$. This quantity is a vector defined as the gradient of the log policy with respect to θ. It quantifies how the policy would change with changes in the different entries in θ. Then, given a positive or a negative reward prediction error, this vector can be used to adjust the policy appropriately [2, 5]. The score function is aggregated over time into the eligibility vector e, subject to an eligibility decay parameter λ. When λ is greater than 0, this permits the algorithm to assign credit for rewards received at the current timestep to actions taken at previous timesteps [4].

3 Momentum in policy-gradient reinforcement learning

3.1 TD(λ) Actor-Critic with Momentum

Since the algorithm described in Section 2.2 implements a form of SGD, it is amenable to improvement using the momentum principle described in Section 2.1. Specifically, we can add momentum to policy-gradient RL by augmenting the update u_t from TD(λ) Actor-Critic with a proportion m of the update from the previous timestep u_{t-1}: $u_t = \eta e_t \delta_t + m u_{t-1}$. This produces a new algorithm, *TD(λ) Actor-Critic with Momentum*.

In addition to helping stabilise learning by filtering out high-frequency noise in parameter updates, another potential advantage of momentum in policy gradient RL is that it may help the algorithm to find global rather than local maxima of reinforcement, or at least to find better local maxima. A limitation of SGD in general is that it is guaranteed only to converge to local optima; this can be especially problematic in RL environments, which are often characterised by non-convex objective functions. In such settings, adding momentum to a policy-gradient RL algorithm might serve to propel the algorithm *past* poor local maxima of reward, and thereby help to produce better overall policies at convergence.

3.2 TD(λ) Actor-Critic with Mood

In animal learning and decision making, one potential impediment to the use of a momentum term is that momentum requires the algorithm to have access to u_{t-1}, the vector of parameter updates at the previous timestep. We aim to show here that a reasonable approximation to u_{t-1} can be constructed using a moving average of reward prediction errors. We are interested in this moving average because of its psychological interpretation as a *mood* variable [10]. Specifically, we follow [10] in defining a mood variable h_t that is recursively updated via a simple error-correcting rule (delta rule) with learning rate η_h and the current prediction error δ_t as a target:

$$h_t = h_{t-1} + \eta_h (\delta_t - h_{t-1}) = \eta_h \sum_{\tau=0}^{t-1} [(1 - \eta_h)^\tau \delta_{t-\tau}] \tag{2}$$

Next, we can unroll the definition of the momentum term and rewrite it as a sum of the products of eligibility traces and prediction errors at previous timesteps:

$$m u_{t-1} = m\eta e_{t-1}\delta_{t-1} + m^2 \eta e_{t-2}\delta_{t-2} + m^3 \eta e_{t-3}\delta_{t-3} + \ldots + m^{t-1}\eta e_1 \delta_1 = \eta \sum_{\tau=1}^{t-1} [m^\tau e_{t-\tau}\delta_{t-\tau}] \tag{3}$$

We now seek to show that this sum of products can be approximated by a moving average of reward prediction errors (that is, by the mood variable h). To this end, we first approximate the past eligibility traces $e_{t-\tau}$ from Equation 3 with the most recent eligibility trace e_{t-1} and move this term outside the sum. Then, by setting the learning rate η_h from Equation 2 to $1 - m$, the mood variable h becomes proportional to the approximated sum in Equation 3. Consequently, mood from trial $t - 1$ can be used to approximate momentum at trial t:

$$m u_{t-1} \approx \eta e_{t-1} \sum_{\tau=1}^{t-1} [m^\tau \delta_{t-\tau}] \approx \eta e_{t-1} \frac{m}{1-m} h_{t-1} \tag{4}$$

Finally, since the approximate momentum update in Equation 4 depends only on quantities available at trial $t-1$, it can be applied in advance at the end of trial $t-1$, rather than waiting until the end of trial t (cf. Nesterov momentum [8]). This produces the *TD(λ) Actor-Critic with Mood* algorithm, which uses a mood variable to help approximate momentum:

TD(λ) Actor-Critic with Mood. The general structure of TD(λ) Actor-Critic is retained, except that a mood variable is calculated on each trial and used to bias parameter updates according to the recent history of reward prediction errors.

$$\delta_t = r + V_{t-1}(s') - V_{t-1}(s) \quad \text{Calculate reward prediction error}$$
$$V_t(s) = V_{t-1}(s) + \alpha\delta_t \quad \text{Update state value}$$
$$h_t = h_{t-1} + (1-m)(\delta_t - h_{t-1}) \quad \text{Update mood}$$
$$e_t = \lambda e_{t-1} + \nabla_\theta \log_{\pi_\theta}(s,a) \quad \text{Increment eligibility trace}$$
$$u_t = \eta e_t \left[\delta_t + \frac{m}{1-m} h_t\right] \quad \text{Calculate parameter update}$$
$$\theta_{t+1} = \theta_t + u_t \quad \text{Update parameters}$$

4 Simulation results

Above, we described one extant policy-gradient RL algorithm (TD(λ) Actor-Critic), and two novel algorithms (TD(λ) Actor-Critic with Momentum, and TD(λ) Actor-Critic with Mood). Here, we assess the simulated performance of these three algorithms in a standard reinforcment learning testbed: the 10-armed bandit problem. Our goal is to determine whether either momentum or mood-approximated momentum help to accelerate learning in this setting.

In the 10-armed bandit problem, the agent can choose from among 10 choice options ('arms'), each of which is characterised by a different mean payout drawn from a unit normal. An agent's task in this environment is to choose arms that maximise the amount of reward that it receives.

We implemented the three algorithms with a softmax policy parameterised by the vector θ, which has length equal to the number of choice options, where the i-th entry of θ denotes strength of preference for the i-th choice option. These preferences can be thought of as analogous to Q-values, in that they denote some measure of the subjective utility of choosing different options; unlike Q-values, however, preferences for different options are not interpretable in terms of expected future return. Each choice option is represented by the feature vector $\phi(a)$, which is a one-hot vector (all 0 except for the entry corresponding to a, which is 1). The exact form of the policy is as below, where A is the set of bandit arms:

$$\pi_\theta(a) = \frac{e^{\phi(a)^T\theta}}{\sum_{\hat{a}\in A} e^{\phi(\hat{a})^T\theta}} \tag{5}$$

With this policy parameterisation, the score function can be expressed in terms of ϕ:

$$\nabla_\theta \log \pi_\theta(a) = \phi(a) - \mathbb{E}_{\pi_\theta}[\phi(\cdot)] \tag{6}$$

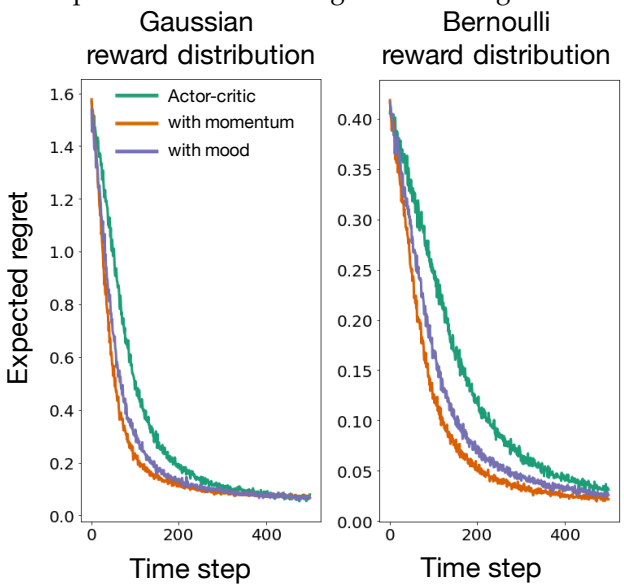


Figure 1: Learning curves (quantified by expected regret averaged across 2000 simulations of 500 timesteps each) on two variants of the 10-armed bandit testbed for three algorithms: TD(λ) Actor-Critic (green), TD(λ) Actor-Critic with Momentum (orange), and TD(λ) Actor-Critic with Mood (purple). Left: testbed with Gaussian payout distribution (payout standard deviation = 1). Right: testbed with Bernoulli payout distribution. In both settings, an algorithm with momentum performs best, followed by an algorithm with mood, followed by an algorithm with neither.

Figure 1 displays the performance of the three algorithms for both Gaussian and Bernoulli payout distributions. Parameters for simulation are: $\lambda = 0.1$, $\gamma = 1$, $\alpha = 0.01$, $\eta = 0.1$, $m = 0.5$. From these results, we can make three primary observations. First, it is clear that a moderate degree of momentum ($m = 0.5$, orange line) accelerates learning performance relative to an equivalent algorithm without momentum (green). Second, TD(λ) Actor-Critic with Mood (purple line), which uses a mood term to approximate momentum, captures much of the benefit of momentum without requiring an explicit representation of previous parameter updates. Third, the relative benefits of momentum are stronger for a Bernoulli payout distribution than a Gaussian. This is because Bernoulli payouts have greater variance than Gaussian payouts, such that individual pieces of feedback are less informative regarding the true gradient of reinforcement. As in SGD, in this context a momentum term allows the algorithm to reduce the variance of updates to the parameters of its policy, and therefore to learn more quickly.

5 Conclusions

It is common practice in optimization by SGD to use a momentum term to accelerate convergence [7, 8, 9]. Here, we provide a proof-of-concept result showing that momentum can also be used to accelerate policy-gradient RL. The underlying reason for this is that momentum helps overcome two limitations of steepest-ascent policy-gradient RL methods: first, momentum can help prevent parameter oscillation in settings where performance is more sensitive to small changes in some parameters than in others (e.g., driving in an urban environment, performance is much more sensitive to small differences in the angle of the wheels than small differences in speed). Second, for on-line RL algorithms in stochastic environments, individual rewards or state transitions may provide very noisy estimates of the true gradient of reinforcment with respect to the parameters of the policy. For instance, a single reward from a bandit that pays out probabilistically gives only a high-variance sample of its underlying reward probability. By averaging parameter updates across multiple points in time, momentum acts as a low-pass filter on this high-variance quantity, and therefore allows parameters to be updated using a more stable (and more accurate) gradient estimate.

We also show that a momentum term can be reasonably well approximated in the policy-gradient RL setting by a *mood* variable, where mood is defined as an exponential moving average of reward prediction errors [10]. This derivation may shed light upon the role of mood in human and animal learning and decision making. For instance, [10] found evidence for a mood-congruent interaction between mood and RL in human participants, such that participants who self-reported high levels of mood instability tended over-value stimuli that they had encountered in a positive mood, and under-value stimuli encountered in a negative mood. A mood model such as TD(λ) Actor-Critic with Mood provides one explanation for this finding, because this algorithm approximates momentum by updating its policy parameters at each timestep in the direction of the *sum* of the current reward prediction error *and* current mood. The effect of this is to boost preferences for stimuli encountered when mood is positive, as [10] observed in participants high in mood instability.

More broadly, the fact that RL algorithms are improved by the addition of either momentum or mood is a reflection of an interesting general property of learning: if an agent consistently receives positive reward prediction errors, this can often be taken as a sign that the policy the agent has lately been following ought to be reinforced. By contrast, the reverse is true for consistent negative reward prediction errors, which might indicate the necessity of altering the current policy. The two policy-gradient RL algorithms that we have derived in this project take advantage of this general property of learning. In the case of TD(λ) Actor-Critic with Momentum, this representation is made explicit in the form of a momentum term; for TD(λ) Actor-Critic with Mood, it is accomplished implicitly by the use of mood to help approximate momentum.

References

[1] J. Li and N. D. Daw, "Signals in human striatum are appropriate for policy update rather than value prediction," *The Journal of Neuroscience*, vol. 31, no. 14, pp. 5504–5511, 2011.

[2] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[3] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

[4] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *American Control Conference (ACC), 2012*, pp. 2177–2182, IEEE, 2012.

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1928–1937, 2016.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.

[7] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[8] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O(\frac{1}{K^2})$," *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1984.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[10] E. Eldar and Y. Niv, "Interaction between emotional state and learning underlies mood instability," *Nature Communications*, vol. 6, p. 6149, 2015.

[11] R. S. Sutton, "Two problems with back propagation and other steepest descent learning procedures for networks," in *Proceedings of the 8th Annual Conference of the Cognitive Science Society, 1986*, pp. 823–832, 1986.

Learning Multi-Agent Communication with Reinforcement Learning

Sushrut Bhalla*
Department of Computer Engineering
University of Waterloo
Waterloo, ON N2L 3G1
sushrut.bhalla@uwaterloo.ca

Sriram Ganapathi Subramanian
University of Waterloo
Waterloo, ON N2L 3G1
s2ganapathisubrmanian@uwaterloo.ca

Mark Crowley
University of Waterloo
Waterloo, ON N2L 3G1
mcrowley@uwaterloo.ca

Abstract

Deep Learning and back-propagation have been successfully used to perform centralized training with communication protocols among multiple agents in a cooperative environment. In this paper, we present techniques for centralized training of Multi-Agent (Deep) Reinforcement Learning (MARL) using the model-free Deep Q-Network (DQN) as the baseline model and communication between agents. We present two novel, scalable and centralized MARL training techniques (MA-MeSN, MA-BoN), which separate the message learning module from the policy module. The separation of these modules helps in faster convergence in complex domains like autonomous driving simulators. A second contribution uses a memory module to achieve a decentralized cooperative policy for execution and thus addresses the challenges of noise and communication bottlenecks in real-time communication channels. This paper theoretically and empirically compares our centralized and decentralized training algorithms to current research in the field of MARL. We also present and release a new OpenAI-Gym environment which can be used for multi-agent research as it simulates multiple autonomous cars driving cooperatively on a highway. We compare the performance of our centralized algorithms to DIAL and IMS based on cumulative reward achieved per episode. MA-MeSN and MA-BoN achieve a cumulative reward of at-least 263% of the reward achieved by the DIAL and IMS. We also present an ablation study of the scalability of MA-BoN and see a linear increase in inference time and number of trainable parameters compared to quadratic increase for DIAL.

Keywords: Multi-Agent Reinforcement Learning; Autonomous Driving; Deep Reinforcement Learning; Multi-Agent Systems

*Contact Author

1 Introduction

Multi Agent Reinforcement Learning (MARL) deals with the problem of learning optimal policies for multiple interacting agents using RL. MARL algorithms can be applied to cooperative and competitive tasks. The main application for cooperative MARL algorithms is in safe multi-agent autonomous driving. Training independent agents in multi-agent cooperative environments leads to instability during training as the environment (consisting of other RL agents) will exhibit a non-stationary model over time.

To overcome the problem of non-stationarity in the training of MARL agents, the current literature proposes the use of centralized training using communication, information sharing or unified memory between the agents [7, 2, 4]. Effective communication between agents in MARL can be trained using backpropagation [7, 2]. Iterative Message Sharing (*IMS*) [7] employs a message sharing protocol where an aggregated message is generated by averaging the messages from all agents. The final policy is computed greedily from the value function which maps the observation z and aggregate message m_{agg} to the state-action value, given by, $\pi = argmax_a V(z, m_{agg})$. Differentiable Inter-Agent Learning (*DIAL*) [2] also trains communication channels, through back-propagation, for sequential multi-agent environments. However, the messages exchanged between the agents are from the past time-steps. This causes a sub-optimal convergence in dynamic environments as we show in our experiments section. Our work differs from these approaches in two ways. (a) We remove the iterative network structure of communication protocol and replace it with a feed-forward neural network, which reduces the complexity during training and increases the expressibility of the message. (b) We use the centralized structure during training only and train a decentralized policy using a memory module for execution as the communication among agents in autonomous driving environment is not guaranteed.

In this paper, we propose two centralized training algorithms for MARL environments using DQN [5] as the baseline. The first approach extends the idea of using communication channels for message sharing as proposed in [2] to multi-agent same discrete time-step communication, where the communication protocol is trained using back propagation [7]. The second approach introduces a broadcast network which generates a single broadcast message for all agents in the environment and thus reduces channel bandwidth and memory requirements of the approach. We also propose a novel method of boostrapping the training of independent memory module alongside our policy network to achieve fully decentralized cooperative policy for execution. We evaluate our methods against current state of the art techniques in MARL on multi-agent autonomous driving environment. We have developed an OpenAI Gym environment [1] which simulates multiple autonomous and adversary cars driving on a highway. We also evaluate our results on two more multi-agent particle environments with a long time to horizon and a cooperative reward structure [6].

2 Methods

Consider a cooperative multi-agent stochastic game G which is modeled by the tuple $G = (X, S, A, T, R, Z, O)$ with N agents, $x \in X$, in the game. The game environment presents states $s \in S$, and the agents observe an observation $z \in Z$. The observation is generated using the function $Z \equiv O(s, x)$ which maps the state of each agent to its private observation z. The game environment is modeled by the joint transition function $T(s, \mathbf{a_i}, s')$ where $\mathbf{a_i}$ represents the vector of actions for all agents $x \in X$. We use the subscript notation i to represent the properties of a single agent x, a bold subscript $\mathbf{i}$ to represent properties of all agents $x \in X$ and $-\mathbf{i}$ to represent the properties of all agents other than x_i. We use the superscript t to represent the discrete time-step. The environment provides with a reward function R, which can be a shared function to enable a cooperative behavior. In Partially Observable Stochastic Games (POSG), the reward function $R : S \times A$ maps each agent's actions a_i to a private reward. In the following paragraphs, we present two methods for centralized training of cooperative policies in MARL domains, which can be extended to a decentralized execution paradigm. All the centralized training algorithms exhibit the property of having separate message generation and policy modules. MA-MeSN and MA-BoN remove the need for iterative message passing, and thus allowing centralized training with a reduced inference time and still achieving a better cooperative policy than previous approaches.

Multi-Agent Message Sharing Network (MA-MeSN): We present a scalable multi-agent network structure which allows the message generation network f' to be optimized using gradients from policy networks of all agents and thus provides better generalization. Evaluation of IMS in [7] mentions that only one agent is communicating in every iteration of the message sharing loop. To eliminate the iterative communication, we propose a centralized training network with communication channels inspired by the work of DIAL [2]. Fig. 1(a) shows the architecture of the MA-MeSN network where agents are sharing messages and computing the final action-value for the same discrete time-step of the environment. We use 3 way communication where the messages generated by $f'_{-\mathbf{i}}$ is also conditioned on the communication from agent x_i. The messages $m_{-\mathbf{i}}$ are conditioned on the full-state of the observable environment $\{z_i^t, z_{-\mathbf{i}}^t\}$, rather than the private observation of each agent. A neural network f'' is used to evaluate the action-values for agent x_i conditioned on its private observation and messages from other agents in the environment $m_{-\mathbf{i}} \equiv f'(z_{-i}^t, f(z_i^t))$.

This approach has two advantages over DIAL. The messages $m_{-\mathbf{i}}^t(z_{-\mathbf{i}}^t, f(z_i^t))$ are conditioned on the entire observable state at time t, as opposed to DIAL, where messages $m_{-\mathbf{i}}^t(z_{-\mathbf{i}}^{t-1})$ are a function of the previous time-step private observa-

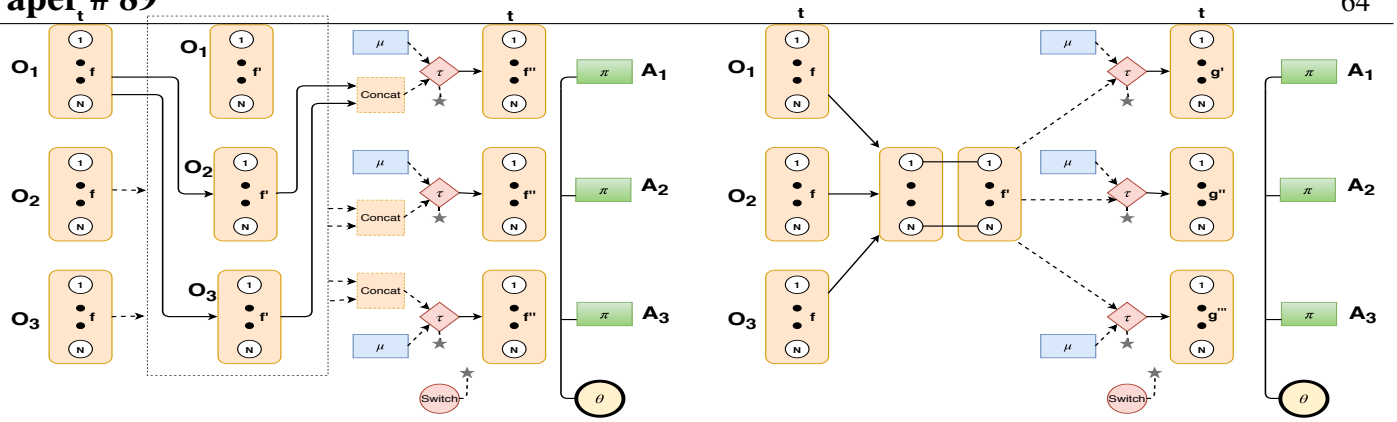


Figure 1: Architecture for Cooperative MARL Network: (a) MA-MeSN (left), (b) MA-BoN (right) with Memory.

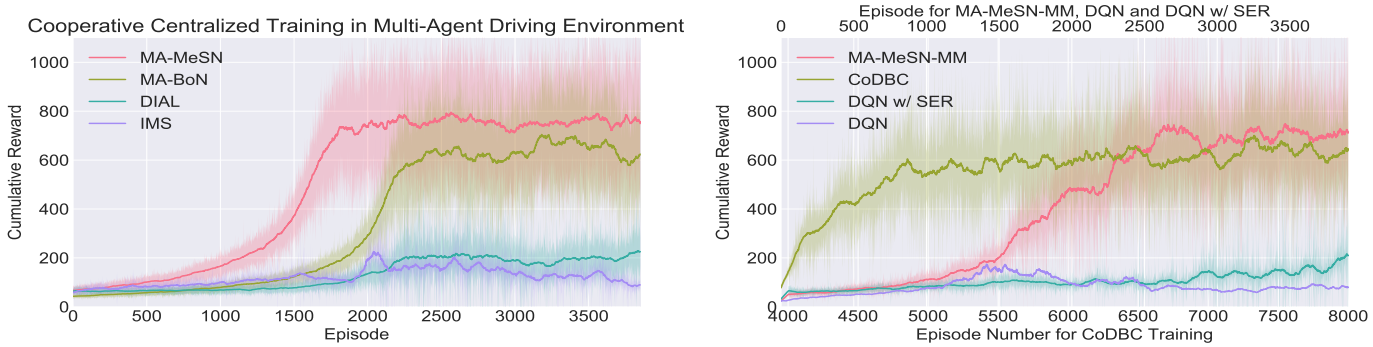


Figure 2: Cumulative reward for (a) Centralized *(left)*, (b) Decentralized *(right)* training on the Driving Environment.

tion of each agent $z_{-\mathbf{i}}^{t-1}$. This results in improved stability in training and a better final cooperative policy. Secondly, MA-MeSN can work with a step-based experience replay buffer with uniform sampling for gradient calculations, whereas, DIAL trains on the samples of current episode to stay current with changing policies of other agents. To achieve decentralized execution using discrete messages, we use Gumbel-Softmax [3] operation on the continuous message generated by each agent [6] during training as it allows for differentiability of the network. Gumbel-Softmax generates a continuous approximation of the categorical distribution by replacing the argmax operation with a Softmax operation. To achieve fully decentralized execution without message sharing, we also propose a LSTM memory module μ associated with each agent's policy network. The $LSTM_\mu$ learns a mapping from agent's private observation to the message generated by agents in the environment which can be used during fully decentralized execution. Thus the individual memory modules along with their policy network can be independently used for fully decentralized execution of cooperative policy by the agents (MA-MeSN-MM).

Multi-Agent Broadcast Network (MA-BoN) A drawback of MA-MeSN is that the network needs to be individually evaluated to compute the action-value for each agent. We note that back-propagation could be used to train a single message ($broadcast \equiv f'(\mathbf{m_i^t})$) which relays a learned representation of all agent's private messages $m_\mathbf{i}^t$. The Fig. 1(b) shows the network architecture for Multi-Agent Broadcast Network (MA-BoN). We deploy a feed-forward broadcast network f' surrounded by a symmetric network structure. The NN $f'(\mathbf{m_i^t})$ learns a combined communication message as the broadcast message. Each agent can now independently evaluate the action-value for their private observation using the function $g'(z_i^t, f'(\mathbf{m_i^t}))$ which is a function of the complete observed state of the environment. This network also allows for parallel action-value evaluations with a single forward pass of the network and avoids the $|P|$ iterations required by IMS and 2 iterations of MA-MeSN. MA-BoN can also be easily decenetralized using discrete messages or by the use of a memory module $LSTM_\pi$ trained in parallel to the policy network (MA-BoN-MM).

3 Experiments and Results

Centralized Training on Multi-Agent Driving Environment We have developed a multi-vehicle driving simulator which simulates multiple autonomous and adversary vehicles driving on a highway. The adversary's objective is to hit the closest car and all cooperative autonomous cars must avoid crashes. The MARL agents receive a hidden observation of the environment and a private reward based on distance from the closest agent but don't know which car is autonomous or adversary. The agents can communicate using discrete limited bandwidth channel.

Table 1: Comparison of scalability in terms of memory and inference time analysis for MA-BoN (ours), DIAL and IMS.

	MA-BoN			**DIAL**			**IMS**		
#Cars	Time (*sec*)	#Param (*M*)	CRPE	Time (*sec*)	#Param (*M*)	CRPE	Time (*sec*)	#Param (*M*)	CRPE
2	13.2658	2.826	512.4408	14.5619	1.761	117.1410	21.4447	2.908	194.4888
3	21.1892	3.383	443.5566	25.1714	2.572	89.25467	31.1567	2.924	191.4618
4	31.1815	3.940	409.5076	37.8002	3.383	79.10172	39.2954	2.941	160.3807
5	41.8882	4.497	376.4217	50.6050	4.194	70.96123	47.6846	2.957	109.6350
6	55.3998	5.054	342.8269	64.0858	5.005	68.8291	55.0327	2.974	93.84936
7	71.1721	5.611	279.1121	80.4211	5.816	62.41487	63.5499	2.990	83.25440

The results for centralized training of cooperative multi-agents (averaged over 20 runs) are shown in Fig. 2(a). The policy is approximated using a neural network with 2 layers of size $[4096, 64]$ for MA-MeSN, MA-BoN and IMS; DIAL uses a neural network of size $[6144, 64]$. All agents have a linearly decaying exploration strategy over the first $100K$ steps. We updated DIAL to sample from a step-based replay buffer with importance sampling which weighs the newer episodes with a higher probability of being sampled. For the IMS algorithm, we arrived at using $P = 5$ for communication iterations through cross-validation. We achieve the highest cumulative reward with MA-MeSN followed by the MA-BoN algorithm. The IMS and DIAL algorithm are able to improve on the policy achieved by independent DQN, due to the shared information using trained communication channels. The MA-BoN and MA-MeSN use step-based replay memory (z_i^t, a_i^t, r_i^t) along with $m_{-\mathbf{i}}^t$ which provides better indexing of the changing policies of other agents over time and thus allows for a more stable training algorithm. The intuition for sharing messages in the current time-step is that it provides other agents with an insight into the future action policy of the environment containing other learning agents. We thus see a stable learning curve with faster convergence properties than DIAL and IMS. The MA-BoN results show comparative performance to MA-MeSN and provides us with the benefit of reduction in the number of communication layers needed from $|N| \times |N|$ to $|N|$; along with reduced inference time for state-value prediction.

Ablation Study of scalability of MA-BoN We demonstrate the scalability of the MA-BoN approach compared to IMS and DIAL through an ablation study where we increase the number of agents in the environment. The table 1 shows a comparison of the inference time to complete an episode (**Time**), the total number of parameters required (**#Param**) and the average cumulative reward per episode (**CRPE**) when the number of agents in the environment is increased. We see that the number of parameters (#*Params*) and communication connections between agents for MA-BoN grows linearly compared to DIAL and thus we also see a slower rate of increase in the number of trainable parameters. The inference time of MA-BON is comparable to IMS and better than DIAL, while achieving better performance than both DIAL and IMS. We use *CRPE* as the measure of performance of the algorithm. *CRPE* are computed by averaging results of 5 training runs of $15,000$ episodes or $2.5M$ steps. MA-BoN outperforms DIAL and IMS by a large margin and also shows better scalability as the message generation network for each agent is optimized using the cumulative gradients from all agents' temporal difference loss. Thus, the message is more generalizable in complex settings. DIAL and IMS use policy gradients to update the current agent's message and policy joint parameters which leads to reduced robustness of the message shared between agents.

Decentralized Cooperative Policy on Multi-Agent Driving Environment Decentralized execution could be achieved by using a discrete channel between agents or by completely removing message sharing between agents. First, we evaluate the performance of MA-MeSN and MA-BoN networks with discrete message sharing. The performance is measured in terms of cumulative reward achieved in each episode compared to fully centralized training with continuous messages. MA-MeSN and MA-BoN networks were able to maintain 98.35% and 86.47% of the performance from centralized policy after 4000 episodes. The distribution of messages in MA-MeSN network varies based on which agent is closer to an adversary; whereas in the MA-BoN the message varies based on the distance of adversary to the any autonomous agent. We compare two distributions generated by two different agents using chi-squared distance. We get a value of 0.07457 and 0.00105 for MA-MeSN and MA-BoN respectively. Thus, the generalizability of broadcast messages in MA-BoN allows us to extend the MA-BoN network to environments with a variable number of agents. We note that the channels between agents can be unreliable and thus agents must be able to execute a cooperative decentralized policy without communication.

Centralized Training - Predator Prey All experimental results (Fig 3(a)) for Predator-Prey domain represent the average over 5 runs and the results were smoothed using a moving average. All algorithms use a linearly decaying exploration schedule for the first $100K$ steps from 1.0 to 0.05 and we then use a constant value of 0.05 for the rest of the training. All experiments are run for $7M$ steps and $60K$ episodes. All agents in the environment use parameter sharing of weights and biases of the neural network with a size of 1 hidden layer with 256 hidden units, with a communication channel of size 8 units. Our centralized training method MA-MeSN was able to achieve good performance in this environment

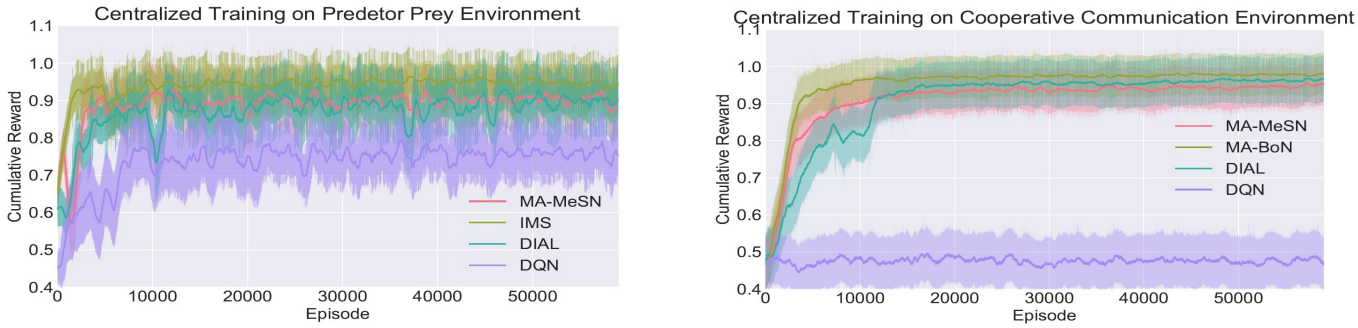


Figure 3: Cumulative reward for Centralized training on (a.) Predator Prey *(left)*, (b.) Cooperative Communication.

as well (very close to IMS). DIAL performs poorly in dynamic environments as the message is generated based on old observations of the environment. The results clearly shows that our algorithms are extendable to environments with sparse cooperative rewards.

Centralized Training - Cooperative Comm. Results in Fig 3(b) for cooperative communication represents an average over 5 runs with the same hyper-parameters as the previous section and for $7M$ steps and $100K$ episodes. To achieve discrete 2-bit communication between the speaker and the listener, we apply a softmax on the output of the speaker before feeding it to the listener. MA-BoN and MA-MeSN don't use parameter sharing and the communication channel from the listener is masked with dummy values for this experiment. Again the superior performance of MA-BoN is seen.

4 Conclusion and Future Work

We have proposed two novel scalable centralized training algorithms (MA-MeSN, MA-BoN) for training multiple autonomous agents in an environment. The MA-MeSN uses idea of iterative message sharing and trains messages using back-propagation. The MA-BoN uses back-propagation to train multiple agents using a single broadcast network which is representative of the full state of the environment. We also propose a method to achieve discrete message decentralized execution and fully decentralized execution using memory module (MA-MeSN-MM). We note that MA-BoN messages are generalizable and robust. Thus as a natural next step, we will extend the centralized training algorithms to environments with varying number of agents.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[2] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[3] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[4] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[6] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[7] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2244–2252. Curran Associates, Inc., 2016.

Count-Based Exploration with the Successor Representation*

Marlos C. Machado†
Google Brain
Montreal, QC, Canada
marlosm@google.com

Marc G. Bellemare
Google Brain
Montreal, QC, Canada
bellemare@google.com

Michael Bowling
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
mbowling@ualberta.ca

Abstract

In this paper we provide empirical evidence showing that the norm of the successor representation (SR), while it is being learned, can be used to generate effective exploration bonuses for reinforcement learning algorithms. The SR is a representation that defines state generalization by the similarity of successor states. In our experiments the agent maximized the reward function $R_t + \beta r_{\text{int}}$, where R_t is the reward signal generated by the environment at time step t, β is a scaling factor, and r_{int} is the exploration bonus such that $r_{\text{int}} = \frac{1}{||\psi(S_t)||_2}$, with $\psi(S_t)$ being the agent's estimate of the SR in state S_t. In the tabular case, when augmenting Sarsa with the proposed exploration bonus, we obtained results similar to those obtained by theoretically sample-efficient approaches. We evaluated our algorithm in traditionally challenging tasks such as RiverSwim and SixArms. We also evaluated this idea in hard-exploration Atari games where function approximation is required. We obtained state-of-the-art performance in a low sample-complexity regime, outperforming pseudo-count-based methods as well as the recently introduced Random Network Distillation (RND). We used a deep neural network to approximate both the value function and the SR. In the extended version of this paper we also provide some theoretical justification to the use of the norm of the SR as an exploration bonus by showing how, while it is being learned, it implicitly keeps track of state visitation counts. We believe this result might lead to a different and fruitful research path for exploration in reinforcement learning.

Keywords: Computational reinforcement learning, successor representation, exploration, function approximation, Atari games.

Acknowledgements

The authors would like to thank Jesse Farebrother for the initial implementation of DQN used in this paper, Georg Ostrovski for the discussions and for kindly providing us the exact results we report for the pseudo-count-based methods, and Yuri Burda for kindly providing us the data we used to compute the performance we report for RND. We would also like to thank Carles Gelada, George Tucker and Or Sheffet for useful discussions, as well as the anonymous reviewers for their feedback. This work was supported by grants from Alberta Innovates Technology Futures and the Alberta Machine Intelligence Institute (Amii). Computing resources were provided by Compute Canada through CalculQuébec.

*This paper is an extended abstract of the following article: "M. C. Machado, M. G. Bellemare, M. Bowling. Count-Based Exploration with the Successor Representation. CoRR abs/1807.11622, 2018."

† work performed while at the Department of Computing Science at the University of Alberta.

1 Introduction

Reinforcement learning (RL) tackles sequential decision making problems by formulating them as tasks where an agent must learn how to act optimally through trial and error interactions with a complex, unknown, stochastic environment. The goal in these problems is to maximize the discounted sum of the numerical reward signal observed at each time step. The actions taken by the agent influence not just the immediate reward it observes but also the future states and rewards it will observe, implicitly requiring the agent to deal with the trade-off between short-term and long-term consequences. Here we focus on the problem of exploration in reinforcement learning, which is the problem of selecting appropriate actions to explore the state space to gather information while taking the aforementioned trade-off into consideration.

Surprisingly, the most common approach in the field is to select exploratory actions uniformly at random, with even high-profile success stories being obtained with this strategy (e.g., [10]). However, random exploration often fails in environments with sparse rewards. In this paper we introduce a novel approach for exploration in reinforcement learning based on the successor representation [5]. The successor representation is a representation that generalizes between states using the similarity between their successors, that is, the states that follow the current state given the agent's policy.

The main contribution of this paper is to show that the norm of the successor representation can be used as an exploration bonus. We demonstrate this empirically in both tabular and function-approximation cases. For the latter we design a deep reinforcement learning algorithm that achieves state-of-the-art performance in hard exploration Atari games when in a low sample-complexity regime. A more thorough discussion about the proposed idea, as well as theoretical results suggesting that the successor representation, while it is being learned, might encode some notion of state visitation counts is available in the extended version of this paper [7].

2 Preliminaries

We consider the traditional reinforcement learning framework. We refer the reader to Sutton and Barto's textbook for a detailed presentation of the basic formalism [14]. In this paper we assume the reader is familiar with the basic concepts in the field of reinforcement learning. The ideas presented here are based on the **successor representation** (SR) [5]. The successor representation with respect to a policy π, Ψ_π, is defined as

$$\Psi_\pi(s, s') = \mathbb{E}_{\pi,p}\Big[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}\{S_t = s'\} \Big| S_0 = s\Big],$$

where $\gamma \in [0, 1)$ and $\mathbb{I}$ denotes the indicator function. This expectation can be estimated from samples with TD learning:

$$\hat{\Psi}(S_t, j) \leftarrow \hat{\Psi}(S_t, j) + \eta\Big(\mathbb{I}\{S_t = j\} + \gamma\hat{\Psi}(S_{t+1}, j) - \hat{\Psi}(S_t, j)\Big), \tag{1}$$

for all $j \in \mathcal{S}$ and η denoting the step-size. The definition of the SR can also be extended to features. Successor features [1] generalize the SR to the function approximation setting. We use the definition for the uncontrolled case in this paper.

Definition 2.1. *For a given $0 \leq \gamma < 1$, policy π, and for a feature representation $\boldsymbol{\phi}(s) \in \mathbb{R}^d$, the successor features for a state s are:*

$$\boldsymbol{\psi}_\pi(s) = \mathbb{E}_{\pi,p}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(S_t) \middle| S_0 = s\right].$$

3 The Norm of the Successor Representation as an Exploration Bonus

It is now well-known that the successor representation incorporates diffusion properties of the environment. These properties can be used to accelerate learning, for example, with options that promote exploration [9]. Inspired by these results, in this section we argue that the successor representation can be used in a more direct way to promote exploration.

To demonstrate the usefulness of the norm of the successor representation as an exploration bonus we compare the performance of traditional Sarsa to Sarsa+SR, an algorithm introduced here that incorporates the norm of the successor representation as an exploration bonus in the Sarsa update. The update equation for Sarsa+SR is

$$\hat{q}(S_t, A_t) \leftarrow \hat{q}(S_t, A_t) + \alpha\left(R_t + \beta\frac{1}{||\hat{\Psi}(S_t)||_2} + \gamma\hat{q}(S_{t+1}, A_{t+1}) - \hat{q}(S_t, A_t)\right), \tag{2}$$

where β is a scaling factor and, at each time step t, $\hat{\Psi}(S_t, \cdot)$ is updated before $\hat{q}(S_t, A_t)$ as per Equation 1.

We evaluted this algorithm in RiverSwim and SixArms [13], traditional domains in the PAC-MDP literature that are used to evaluate provably sample-efficient algorithms. In these domains it is very likely that an agent will first observe a small

Table 1: Comparison between Sarsa and Sarsa+SR. A 95% confidence interval is reported between parentheses.

	Sarsa		Sarsa + SR	
RIVERSWIM	26,526	(2,350)	1,989,479	(167,189)
SIXARMS	284,013	(88,511)	2,625,132	(516,804)

reward generated in a state that is easy to get to. If the agent does not have a good exploration policy it is likely to converge to a suboptimal behavior, never observing larger rewards available in states that are difficult to get to.

Our results suggest that the proposed exploration bonus has a profound impact in the algorithm's performance. When evaluating the agent for $5{,}000$ time steps, Sarsa obtains an average return of approximately $26{,}000$, while Sarsa+SR obtains an approximate average return of 2 million! Notice that, in RIVERSWIM, the reward that is "easy to get" has value 5, implying that, different from Sarsa+SR, Sarsa almost never explores the state space well enough. The actual numbers, which were averaged over 100 runs, are available in Table 1. Details about the task, parameters used, as well as the empirical methodology are available in the extended version of this paper [7].

4 Counting Feature Activations with the SR

In large environments, where enumerating all states is not an option, directly using Sarsa+SR as described in the previous section is not viable. Using neural networks to learn a representation while learning to estimate state-action value function is the approach that currently often leads to state-of-the-art performance in the field. However, learning the SR becomes more challenging when the representation, ϕ, is also being learned. In this section we describe an algorithm that uses the same ideas described so far but in the function approximation setting. Our algorithm was inspired by recent work that have shown that successor features can be learned jointly with the feature representation itself [6, 9].

An overview of the neural network we used to learn the agent's value function while also learning the feature representation and the SR is depicted in Figure 1. The layers used to compute the state-action value function, $\hat{q}(S_t, \cdot)$, are structured as in DQN [10], but with different numbers of parameters (i..e, filter sizes, stride, and number of nodes). This was done to match Oh et al.'s architecture, which is known to succeed in the auxiliary task of predicting the agent's next observation, which we detail below [11]. From here on, we call the part of our architecture that predicts $\hat{q}(S_t, \cdot)$ DQN$_e$ to distinguish between the parameters of this network and DQN. It is trained to minimize the mixed Monte-Carlo return (MMC), which has been used in the past by the algorithms that achieved succesful exploration in deep reinforcement learning [3, 12]. The reward signal the agent maximizes is $R_t + \beta r_{\text{int}}$, where R_t denotes the reward signal generated by the environment and r_{int} denotes the exploration bonus obtained from the successor features of the internal representation, ϕ, which will be defined below. Moreover, to ensure all features are in the same range, we normalize the feature vector so that $||\phi(\cdot)||_2 = 1$. In Figure 1 we highlight with ϕ the layer in which we normalize its output. Notice that the features are always non-negative due to the use of ReLU gates.

The successor features, $\psi(S_t)$, at the bottom of the diagram, are obtained by minimizing the loss

$$\mathcal{L}_{\text{SR}} = \mathbb{E}_{\pi,p}\Big[\big(\phi(S_t;\theta^-) + \gamma\psi(S_{t+1};\theta^-) - \psi(S_t;\theta)\big)^2\Big].$$

Zero is a fixed point for the SR, which is particularly concerning in settings with sparse rewards. The agent might end up learning to set $\phi(\cdot) = \vec{0}$ to achieve zero loss. We address this problem by not propagating $\nabla\mathcal{L}_{\text{SR}}$ to ϕ (this is depicted in Figure 1 as an open circle stopping the gradient). The distinction between θ and θ^- is standard in the field, with θ^- denoting the parameters of the target network, which is updated less often for stability purposes [10]. We also create an auxiliary task to encourage a representation to be learned before a non-zero reward is observed. As Machado et al. [9], we use the auxiliary task of predicting the next observation, learned through the architecture proposed by Oh et al. [11], which is depicted as the top layers in Figure 1. The loss we minimize for this last part of the network is

$$\mathcal{L}_{\text{Recons}} = \big(\hat{S}_{t+1} - S_{t+1}\big)^2.$$

The overall loss minimized by the network is $\mathcal{L} = w_{\text{TD}}\mathcal{L}_{\text{TD}} + w_{\text{SR}}\mathcal{L}_{\text{SR}} + w_{\text{Recons}}\mathcal{L}_{\text{Recons}}$.

The last step in describing our algorithm is to define r_{int}, the intrinsic reward we use to encourage exploration. We choose the exploration bonus to be the inverse of the ℓ_2-norm of the vector of successor features of the current state, as in Sarsa+SR. That is,

$$r_{\text{int}}(S_t;\theta^-) = \frac{1}{||\psi(S_t;\theta^-)||_2},$$

where $\psi(S_t;\theta^-)$ denotes the successor features of state S_t parametrized by θ^-. The exploration bonus comes from the same intuition presented in the previous section (we observed in preliminary experiments not discussed here that DQN performs better when dealing with positive rewards).

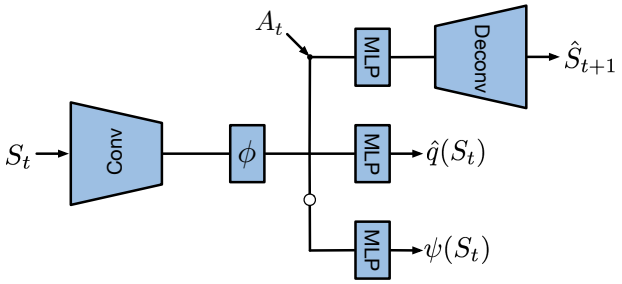


Figure 1: Neural network architecture used by our algorithm when learning to play Atari games.

5 Evaluation of Exploration in the Deep Reinforcement Learning Case

We evaluated our algorithm on the Arcade Learning Environment [2]. Following Bellemare et al.'s taxonomy [3], we focused on the Atari games with sparse rewards that pose hard exploration problems. We used the evaluation protocol proposed by Machado et al. [8]. We used the game MONTEZUMA'S REVENGE to tune our parameters. The reported results are the average over 10 seeds after 100 million frames. We evaluated our agents in the stochastic setting (sticky actions, $\varsigma = 0.25$) using a frame skip of 5 with the full action set. The agent uses the game screen as input.

Our results were obtained with the algorithm described in Section 4. We set $\beta = 0.025$ after a rough sweep over values in the game MONTEZUMA'S REVENGE. We annealed ϵ in DQN's ϵ-greedy exploration over the first million steps, starting at 1.0 and stopping at 0.1 as done by Bellemare et al. [3]. We trained the network with RMSprop with a step-size of 0.00025, an ϵ value of 0.01, and a decay of 0.95, which are the standard parameters for training DQN [10]. The discount factor, γ, is set to 0.99 and $w_{\text{TD}} = 1$, $w_{\text{SR}} = 1000$, $w_{\text{Recons}} = 0.001$. The weights w_{TD}, w_{SR}, and w_{Recons} were set so that the loss functions would be roughly the same scale. All other parameters are the same as those used by Mnih et al. [10].

Table 2 summarizes the results after 100 million frames. The performance of other algorithms is also provided for reference. Notice we are reporting *learning performance* for all algorithms instead of the maximum scores achieved by the algorithm. We use the superscript $^{\text{MMC}}$ to distinguish between the algorithms that use MMC from those that do not. When comparing our algorithm, DQN$_e^{\text{MMC}}$+SR, to DQN we can see how much our approach improves over the most traditional baseline. By comparing our algorithm's performance to DQN$^{\text{MMC}}$+CTS [3] and DQN$^{\text{MMC}}$+PixelCNN [12] we compare our algorithm to established baselines for exploration that are closer to our method. By comparing our algorithm's performance to Random Network Distillation (RND) [4] we compare our algorithm to one of the most recent papers in the field with state-of-the-art performance.

As mentioned in Section 4, the parameters of the network we used are different from those used in the traditional DQN network, so we also compared the performance of our algorithm to the performance of the same network our algorithm uses but without the additional modules (next state prediction and SR) by setting $w_{\text{SR}} = w_{\text{Recons}} = 0$ and without the intrinsic reward bonus by setting $\beta = 0.0$. The column labeled DQN$_e^{\text{MMC}}$ contains the results for this baseline. This comparison allows us to explicitly quantify the improvement provided by the proposed exploration bonus.

We can clearly see that our algorithm achieves scores much higher than those achieved by DQN, which struggles in games that pose hard exploration problems. Moreover, by comparing DQN$_e^{\text{MMC}}$+SR to DQN$_e^{\text{MMC}}$ we can see that the provided exploration bonus has a big impact in the game MONTEZUMA'S REVENGE, which is probably known as the hardest game among those we used in our evaluation, and the only game where agents do not learn how to achieve scores greater than zero with random exploration. Interestingly, the change in architecture and the use of MMC leads to a big improvement in games such as GRAVITAR and VENTURE, which we cannot fully explain. However, notice that the change in architecture does not have any effect in MONTEZUMA'S REVENGE. The proposed exploration bonus seems to be essential in games with very sparse rewards. We also compared our algorithm to DQN$^{\text{MMC}}$+CTS and DQN$^{\text{MMC}}$+PixelCNN. We can observe that, on average, DQN$_e^{\text{MMC}}$+SR outperforms these algorithms while being simpler since it does not require a density model. Instead, our algorithm requires the SR, which is domain-independent as it is already defined for every problem since it is a component of the value function estimates [5].

Finally, DQN$_e^{\text{MMC}}$+SR also outperforms RND [4] when it is trained for 100 million frames. Importantly, RND is currently considered to be the state-of-the-art approach for exploration in Atari games. Burda et al. did not evaluate RND in FREEWAY, thus we do not report any scores for RND in this game.

A more thorough analysis of the impact of the different components of the proposed algorithm (the importance of the auxiliary task, the impact of using a different p-norm of the SR, among other things) is available in the extended version of this paper [7]. It also contains the learning curves of these algorithms and their performance after different amounts of experience.

Table 2: Performance of the proposed algorithm, $\text{DQN}_e^{\text{MMC}}$+SR, compared to various agents on Atari games. The DQN results reported are from Machado et al. [8] while the DQN^{MMC}+CTS and DQN^{MMC}+PixelCNN results were extracted from Ostrovski et al.'s work and RND results were extracted from Burda et al.'s work. $\text{DQN}_e^{\text{MMC}}$ is another baseline used in the comparison. When available, standard deviations are reported between parentheses.

	DQN	$\text{DQN}_e^{\text{MMC}}$	DQN^{MMC}+CTS	DQN^{MMC}+PixelCNN	RND	$\text{DQN}_e^{\text{MMC}}$+SR
Freeway	32.4 (0.3)	29.5 (0.1)	29.2	29.4	- -	29.5 (0.1)
Gravitar	118.5 (22.0)	1078.3 (254.1)	199.8	275.4	790.0 (122.9)	430.3 (109.4)
Mont. Rev.	0.0 (0.0)	0.0 (0.0)	2941.9	1671.7	524.8 (314.0)	1778.6 (903.6)
Private Eye	1447.4 (2,567.9)	113.4 (42.3)	32.8	14386.0	61.3 (53.7)	99.1 (1.8)
Solaris	783.4 (55.3)	2244.6 (378.8)	1147.1	2279.4	1270.3 (291.0)	2155.7 (398.3)
Venture	4.4 (5.4)	1220.1 (51.0)	0.0	856.2	953.7 (167.3)	1241.8 (236.0)

6 Conclusion

RL algorithms tend to have high sample complexity, which often prevents them from being used in the real-world. Poor exploration strategies is one of the main reasons for this high sample-complexity. Despite all of its shortcomings, uniform random exploration is, to date, the most commonly used approach for exploration. This is mainly due to the fact that most approaches for tackling the exploration problem still rely on domain-specific knowledge (e.g., density models, handcrafted features), or on having an agent learn a perfect model of the environment. In this paper we introduced a general method for exploration in RL that implicitly counts state (or feature) visitation in order to guide the exploration process. It is compatible to representation learning and the idea can also be adapted to be applied to large domains.

References

[1] André Barreto, Will Dabney, Rémi Munos, Jonathan Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Successor Features for Transfer in Reinforcement Learning. In *NIPS*, 2017.

[2] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[3] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In *NIPS*, pages 1471–1479, 2016.

[4] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. In *ICLR*, 2019.

[5] Peter Dayan. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624, 1993.

[6] Tejas D. Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J. Gershman. Deep Successor Reinforcement Learning. *CoRR*, abs/1606.02396, 2016.

[7] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-Based Exploration with the Successor Representation. *CoRR*, abs/1807.11622, 2018.

[8] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[9] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption Discovery through the Deep Successor Representation. In *ICLR*, 2018.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level Control through Deep Reinforcement Learning. *Nature*, 518:529–533, 2015.

[11] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder P. Singh. Action-Conditional Video Prediction using Deep Networks in Atari Games. In *NIPS*, pages 2863–2871, 2015.

[12] Georg Ostrovski, Marc G. Bellemare, Aaron van den Oord, and Rémi Munos. Count-Based Exploration with Neural Density Models. In *ICML*, pages 2721–2730, 2017.

[13] Alexander L. Strehl and Michael L. Littman. An Analysis of Model-based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[14] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

An empirical evaluation of Reinforcement Learning Algorithms for Time Series Based Decision Making

Alberto C. Chapchap
alberto.chapchap@gscap.com.br

Andre Lawson
andre.lawson@gscap.com.br

Dimas Ramos
dimas.ramos@gscap.com.br

Abstract

In this article, an empirical investigation of several tabular reinforcement learning algorithms is carried out for the problem of time series decision making with low signal to noise ratio, focusing on the financial domain. Departing from the empirical finance literature, the main question asked is whether reinforcement learning agents can learn (or hopefully outperform) the reported heuristics in an online fashion. In this context, the performance of temporal difference methods (Q-Learning, Sarsa, Expected Sarsa and Value Function prediction based methods) are evaluated and benchmarked against a widely used strategy from empirical finance. Our contribution is twofold, namely: the empirical evaluation carried out indicates that, when presented with data, the algorithms are able to discover some typical heuristics that have long been reported in the related literature e.g: momentum and mean reversion but conditioned on the current state; therefore, an interesting hybrid dynamic behaviour emerges in the value function estimation and the Q values of the actions. Our second contribution is to note that in this particular setting (small number of discrete actions), the updates of the Q values at each time step can actually be performed for all the possible actions and not only for the action the agent took on that state, leading to a full exploitation behaviour. Across the board, the results using a real world data set suggests that all the tabular methods tested perform better than the strategies reported in the empirical finance literature as well as long only based strategies.

Keywords: Reinforcement Learning, finance, trading, online learning

Acknowledgements

We are deeply indebted to Google DeepMind and the Weinberg Institute for Cognitive Science for their generous support of RLDM2019.

1 Introduction

Decision making given a noisy time series turns out to be a very common problem in many areas of research. In the financial domain, on one hand the efficient market hypothesis (EMH) [1] states that there is no long term structure in the time series itself. On the other hand, the empirical finance literature has reported evidence of two major anomalies, namely: momentum [2] and mean reversion (well surveyed in [3] under the follow the loser heuristic). In case some of these anomalies are present, tabular reinforcement learning agents should be able to learn some of this structure (if any) from the data in a model free fashion.

Applications of the RL framework to financial time series is not a new subject and the literature is vast. For instance, in [4] the problem of learning to trade (i.e whether to buy or sell a particular financial instrument) is investigated and two algorithms, one based on policy search and another based on Q-learning are compared. [5] applied tabular methods to the problem of finding a policy that leads towards an optimized execution strategy. [6] compares three automated stock trading agents, one based on SARSA and two others: one based on trend following (or momentum) heuristics and another based on the combination of momentum and market making. In [7], the work of [4] is extended to a deep architecture where the problem of learning good representations via autoencoders and act optimally is tackled from an end to end fashion. Interestingly, the results among these publications are somewhat mixed, with some cases of successful RL agents, others more in line with the EMH (no structure found at all) and others arguing more in favour heuristic approaches (due to the low signal to noise ratio, learning is too difficult and a good prior might be the best one can do). Arguably, the success of a RL agent is intimately related to the information encoded in the state description and this may be a paramount ingredient in the failure/ success of applications.

That said, the aim of this work is not to propose new state representations for the problem at hand, but rather try to give insight to the following question: under essentially the same state representations used in the empirical finance literature, what is the performance obtained across various tabular RL algorithms ?

The paper is organized as follows: section 2 gives a brief introduction to the reinforcement learning problem. Section 3 describes the algorithms used on the evaluation. Section 4 describes the experimental set up and its corresponding results. In section 5, the present work is concluded.

2 Problem Formulation

In this section, a brief formulation of the problem is presented and the reader is referred to [8] for a thorough presentation. In the general setting, the reinforcement learning problem can be defined by a tuple (**S**, **A**, **P**(R = r ,S'=s' | S=s,A =a , $\mathbf{R}_s^a$, γ, λ). Concretely, at each time step t_i the agent receives a state representation of the environment $S \in \mathbf{S}$, interacts with it by choosing an action $A \in \mathbf{A}$ and receives a stochastic reward $R(S, A) \in \mathbf{R}_s^a$, from the action it has taken in state S and trasitions to a new state S'. The rewards and transitions that the agent incurs are sampled from **P**(R = r ,S'=s' | S=s,A =a). The parameters γ, λ represent the discount factor and the eligibility trace respectively. Based on the experiences from the interaction with the environment, the agent tries to select its new action in order to maximize the expectation of the sum of the discounted future rewards from that state onwards. Hence, setting $G_t = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k}$, the main idea is to find a policy $\pi(A|S)$, a probability distribution over actions given states such that the action-value function, $q_\pi(s, a) = \mathbb{E}_\pi(G_t|S = s, A = a)$, is maximized when the agent takes action A in state S and then follows π afterwards. In this set up, $\mathbb{E}_\pi(.)$ is the expected value of a random variable under the probability distribution **P**(R = r ,S'=s' | S=s,A =a) when the agent samples its actions according to π. Furthermore, the Markovian property allows the expectation to be written in a recursive fashion as the expected Bellman equation i.e:

$$q_\pi(s, a) = \mathbb{E}_\pi \left(R(S, A) + \gamma \sum_{A'} \pi(A'|S')Q(S', A')|S = s, A = a\right). \tag{1}$$

If a policy is given, the value function is the expected return of starting from state S and following the policy π onwards, i.e: $v_\pi(s) = \mathbb{E}_\pi(G_t|S = s)$. In this case the corresponding expected Bellman equation for $V^\pi(S)$ is given by:

$$v_\pi(s) = \mathbb{E}_\pi \left(R(S, \pi(S)) + \gamma v_\pi(s')|S = s\right). \tag{2}$$

In order to solve the reinforcement learning problem, instead of just predicting the values of $q_\pi(s, a)$ the aim is to find the optimal action-value function i.e: $q_*(s, a) = \max_\pi q_\pi(s, a)$, which specifies the best possible performance in the problem at hand. In addtion, $q_*(s, a)$ induces an optimal policy by selecting actions in a greedy fashion. The optimal action-value function also satisfies the the Bellman optimality equation and can be written as:

$$q_*(s, a) = \mathbb{E}\left(R(S, A) + \gamma \max_{a'} q_*(S', a')|S = s, A = a\right). \tag{3}$$

The algorithms proposed in the next section try to solve equations 2 and 3 for the case of value function prediction and control respectively. Moreover, for the problem of prediction the notion of eligibility trace, unifying the backward and forward view methods from [9] is added and evaluated empirically in table 1.

3 Algorithmic Solutions

Before the first algorithm is introduced some particular aspects of finance are going to be outlined in the hope they can be used in a constructive way by the agent. Assuming the agent is able to trade only a fixed amount of contracts of the asset at hand (the bet size is fixed), there are really only 3 possible actions, buying (going long, +1), selling (going short, -1) or staying neutral (doing nothing, 0). Furthermore, without taking into account transactions costs, the rewards are symmetrical (what the agent gets for being short is the opposite of what it gets for holding a long position). Therefore in order to start in a simple fashion, one could try to flatten the reinforcement learning problem into a prediction problem by fixing a policy (say going long). This way the problem now becomes to predict the value function at each state the agent visits, go long if it is positive, short if it is negative and staying neutral otherwise. In order to estimate the value function using the temporal difference method, algorithm 1 from [8], is used.

Algorithm 1 Value Function Prediction

Hyperparameters: α, γ, λ
Initialize V(S) to 0.0
E(S) = 0, for all s in S:
for $t_1, ..., t_n$ **do**
    A := action given by π (in this case π = 1)
    Take action A, observe reward R, and next state S′
    $\delta := R + \gamma V(S') - V(S)$
    $E(S) := (1 - \alpha)E(S) + 1$ (dutch traces)
    for all s ∈ **S do**:
        $V(s) := V(s) + \alpha\delta E(s)$
        $E(s) := \gamma\lambda E(s)$
    end for
    $S := S'$
end for

Next instead of predicting the value function for a fixed policy in each state, the control problem is tackled. The idea now is to predict state-actions pairs and from there derive the optimal policy by acting ϵ - greedy. In addition, it turned out, that the performance of Algorithm 1 is robust for different values of the eligibility trace parameter λ (Table 1). Therefore, in order to keep the number of hyper parameters to a minimum, both Q-learning, SARSA and Expected SARSA Algorithms are investigated by implicitly setting $\lambda = 0.0$ and using ϵ-greedy exploration scheme (Table 2 and 3). In this context, Q-learning ([8] page 131) , SARSA and expected SARSA ([8] page 133), are investigated in an out of the shelf fashion for the case of two actions (long and short).

Algorithm 2 can be viewed as a particular extension of Q-learning, with two basic modifications, namely: first, instead of updating only the state-action pair of the action the agent took, the Q values of all 3 actions (long, short or neutral) in that state are all updated; secondly, given that for each visited state all actions are updated, the agent can follow a fully explotiation policy and act greedily. This is possible because in a given state both the rewards and the next state for every action choosen can be observed by the agent.

Algorithm 2 Modified Q-Learning

Hyperparameters: α, γ
Initialize Q(S, A) to 0.0
for $t_1, ..., t_n$ **do**
    A := $argmax_a Q(S, a)$
    Take action A, observe reward R
    for all $A_i \in$ **A do**:
        Pretend action A_i was taken, observe reward R_i, and next state S_i'
        $\delta_i := R_i + \gamma\ argmax_a Q(S_i', a) - Q(S_i, A_i)$
        $Q(S_i, A_i) := Q(S_i, A_i) + \alpha\delta_i$
    end for
end for

4 Empirical Evaluation

In this section the algorithms are applied to a real world data set comprising the daily closing prices of the SP 500 index from 10 March 1983 to the 31st December 2018. The results are benchmarked against a widely used momentum strategy and a long only strategy: the long/short (resp. long/neutral) momentum strategy compares the price of today with the

moving average of price of the last 180 trading days; in case the price is above a long position (buy the index) is held; conversely, in case it is below the average a short (neutral resp.) position is held; otherwise the position is kept neutral. On the other hand, the long only strategy holds the index forever, aiming at long term capital appreciation.

In order to compare the strategies two metrics are addressed, namely: the cumulative return i.e G_t and the annualized Sharpe ratio [10], i.e $S = \sqrt{252}\frac{\hat{r}}{\hat{\sigma_r}}$, where $\hat{r}$ and $\hat{\sigma_r}$ are the sample average and standard deviation of returns. The main idea behind the Sharpe ratio is to address the return on risk adjusted basis. In addition, in order to make a "fair" comparison the state representation of the tabular algorithms is constructed by calculating the Z score of the prices on the last 180 trading days (same window used by the momentum strategy) and discretizing the score, yielding 8 states which are depicted in x axis of figure 1. The rewards are calculated by multiplying the action took (i.e -1, 0 or +1) at time t by the return of the index at time $t + 1$.

In figure 1 the value function value predicted by algorithm is 1 is reported in basis points (i.e one hundredth of one percent). It is interesting to note that during the years there is some consistency in the value function prediction in each state. Furthermore, a hybrid behavior between momentum and mean reversion emerges, e.g: in the state where the price falls below -3σ a momentum strategy would go short whereas a mean reverting strategy would go long, the RL agent would be long in this case. On the other hand, for price movements in range of $(1\sigma, 2\sigma)$, the RL agent would be long (so would be a momentum strategy whereas a mean reverting strategy would be short). Therefore the behaviour captured is not only dynamic in time but hybrid across the most common heuristics of the literature.

Table 1: Analysis of the impact of different values of discount factor, γ, and elegibility trace, λ, on the Sharpe ratio of Algorithm [1]. The results suggest that $\gamma > 0$ is not necessarily better, since the performance of a myopic agent turns out to be competitive, this could be related to the low signal to noise ratio of the series but needs further investigation.

$\lambda \mid \gamma$	0.00	0.20	0.40	0.60	0.80	0.90	0.99
0.00	0.68	0.67	0.66	0.66	0.68	0.68	0.67
0.20	0.68	0.71	0.72	0.68	0.62	0.58	0.62
0.40	0.68	0.74	0.66	0.66	0.64	0.58	0.55
0.60	0.68	0.67	0.66	0.58	0.45	0.46	0.40
0.80	0.68	0.65	0.65	0.49	0.40	0.20	0.27
0.90	0.68	0.65	0.60	0.47	0.22	0.20	0.25
1.00	0.68	0.66	0.55	0.38	0.24	0.22	0.24

In Tables 2 and 3 the performance of different RL algorithms is compared and benchmarked against the momentum and long only strategies. The results reported are averages and standard deviations accross 18 runs. Interestingly, there is not much variability accross the performance of each algorithm and with a decent amount of confidence, say at least in the 2σ region, all RL agents outperform the benchmark strategies in terms of Sharpe ratio.

5 Conclusions

In this work an empirical study of several tabular RL algorithms has been carried out for the problem of time series based decision making. The experiments on a SP500 data set suggest that the RL agents trained online can outperform some of the widely used heuristics in the finance literature by learning a hybrid dynamic strategy, conditioned on the state representation. Furthermore, for this particular problem, it turned out that a simple value function prediction and a fully exploitation modified Q learning agent are able to provide competitive performance across different algorithms. That said, the performance of the algorithms is also relatively robust for different discount factors γ, suggesting that long term planning does not play a major role in this task. In particular for case of value function prediction, the performance of the algorithm is also investigated for different values of eligibility trace parameters λ and γ.

6 References

[1] Fama, E. F. (1991) Efficient Capital markets: II, *Journal of Finance*, **46**(5), 1575-1617.

Table 2: Cumulative returns, G_0, of the algorithms tested for different values of γ compared with the heuristic baselines.

Performance Table (G_t)	$\gamma = 0.0$	$\gamma = 0.3$	$\gamma = 0.6$	$\gamma = 0.9$	$\gamma = 0.99$
Value Function Prediction	2.04	2.00	1.98	2.02	2.01
Q Learning (2 actions, ϵ =0.01)	1.94 ± 0.10	1.98 ± 0.10	2.04 ± 0.10	2.00 ± 0.15	2.03 ± 0.09
Expected Sarsa (2 actions, ϵ =0.01)	2.02 ± 0.13	1.95 ± 0.14	1.90 ± 0.14	2.03 ± 0.13	2.03 ± 0.15
Sarsa (2 actions, ϵ =0.01)	1.94 ± 0.16	1.97 ± 0.18	1.94 ± 0.18	1.92 ± 0.13	2.03 ± 0.14
Modified Qlearning (3 actions ϵ =0.0)	2.04	2.04	2.04	2.04	2.04
Momentum Long/Short	0.98				
Momentum Long/Neutral	1.93				
Long Only	1.39				

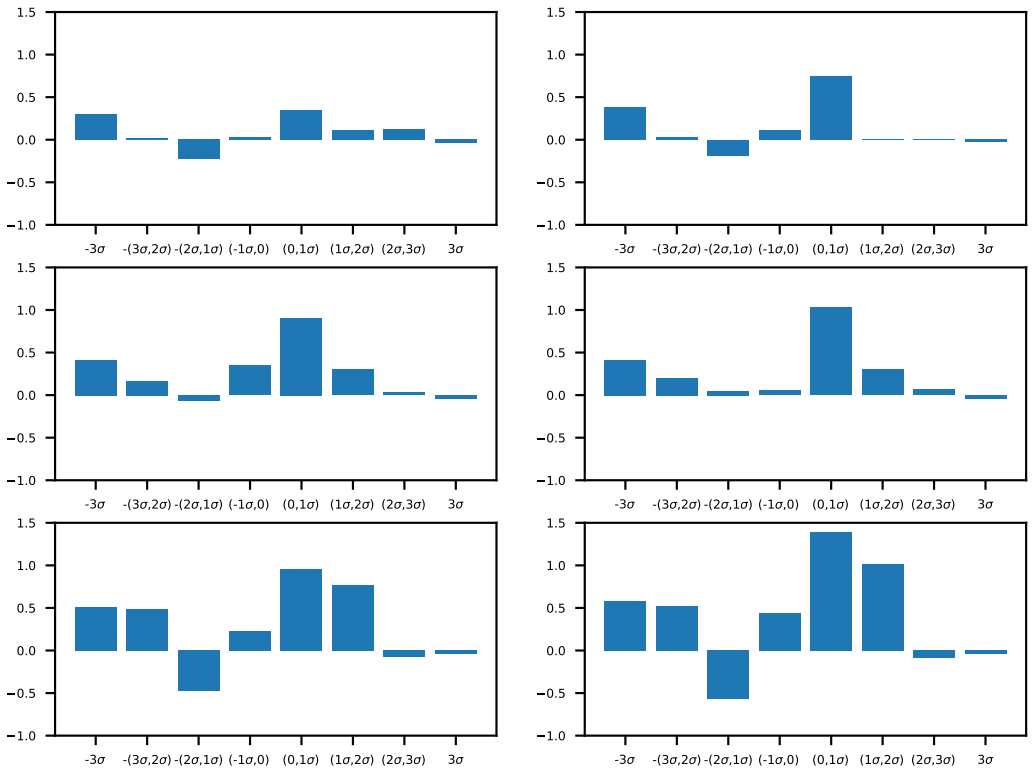


Figure 1: The y-axis shows the estimation of V^{π}, in basis points, from Algorithm [1] for every state in the x-axis. In order to visualize the dynamics time is roughly spaced in 6 year intervals, so the top left is a snapshot of V^{π} on the 15th Feb 89; on the top right on 23rd Jan 95; on the middle left the estimation corresponds to the 22ond Jan 2000; the middle right displays what V^{π} looked on the 28th Sep 2006. The bottom left and right correspond to the 7th Sep 2012 and 21st Aug 2018 respectively.

Table 3: Sharpe ratio,S, of the algorithms tested for different values of γ compared with the heuristic baselines.

Performance Table (Sharpe ratio)	$\gamma = 0.0$	$\gamma = 0.3$	$\gamma = 0.6$	$\gamma = 0.9$	$\gamma = 0.99$
Value Function Prediction	0.68	0.68	0.66	0.68	0.67
Q Learning (2 actions, ϵ =0.01)	0.65 ± 0.03	0.66 ± 0.03	0.68 ± 0.03	0.67 ± 0.05	0.67 ± 0.03
Expected Sarsa (2 actions, ϵ =0.01)	0.68 ± 0.04	0.65 ± 0.05	0.64 ± 0.05	0.68 ± 0.04	0.69 ± 0.05
Sarsa (2 actions, ϵ =0.01)	0.65 ± 0.05	0.66 ± 0.06	0.65 ± 0.06	0.64 ± 0.05	0.69 ± 0.04
Modified Qlearning (3 actions ϵ =0.0)	0.68	0.68	0.68	0.68	0.68
Momentum Long/Short	0.15				
Momentum Long/Neutral	0.47				
Long Only	0.46				

[2] Jegadeesh, N. and Titman, S. (2001) Profitability of momentum strategies: An evaluation of alternative explanations. *Journal of Finance* **56**(2), 699-720.

[3] Li B, Hoi S.C.H (2014) Online portfolio selection: a survey. *ACM Computing Surveys* **46**(3).

[4] Moody, J. E. & Saffell, M. (2001) Learning to Trade via Direct Reinforcement. *IEEE Transactions on Neural Networks* Vol 12, No 4.

[5] Nevmyvaka, Y., Feng, Y. & Kearns, M. (2006) Reinforcement learning for optimized trade execution. *In 23rd International Conference on Machine learning*, pages 673 - 680.

[6] Shrestov, A. & Stone, P. (2004) Three Automated Stock-Trading Agents: A Comparative Study. *Agent Mediated Eletronic Commerce VI: Theories for and Engineering of Sistributed Mechanisms and Systems AMEC 2004* , Lectures Notes in Artificial Intelligence, pp. 194-205, Springer Verlag. Berlin, 2005.

[7] Deng, Y., Kong, K., Ren, Z. & Dai, Q. (2016) Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *Image Processing IEEE Transactions* Vol 25, pp. 4209-4221.

[8] Sutton, R. S. & Barto, A. G. (2018) *Reinforcement Learning: An Introduction, 2nd Edition, Near final Draft - May, 27, 2018* MIT Press

[9] van Seijen, H. & Sutton, R. S. (2014) True online TD(λ). *In Proceedings of the 31st International Conference on Machine Learning*, pages: 692 - 700.

[10] Sharpe, W. (1994) The Sharpe Ratio. *The Journal of Portfolio Management, 21 (1) 49 - 58.*

Temporal Abstraction in Cooperative Multi-Agent Systems

Jhelum Chakravorty
Department of Computer Science
McGill University/Mila
Montreal
jhelum.chakravorty@mail.mcgill.ca

Sumana Basu
Department of Computer Science
McGill University/Mila
Montreal
sumana.basu@mail.mcgill.ca

Andrei Lupu
Department of Computer Science
McGill University/Mila
Montreal
andrei.lupu@mail.mcgill.ca

Doina Precup
Department of Computer Science
McGill University/Mila
Montreal
dprecup@cs.mcgill.ca

Abstract

In this work we introduce temporal abstraction in cooperative multi-agent systems (or *teams*), which are essentially *decentralized Markov Decision processes* (Dec-MDPs) or *dec. Partially Observable MDPs* (Dec-POMDPs). We believe that as in the case of single-agent systems, *option* framework gives rise to faster convergence to the optimal value, thus facilitating transfer learning.

The decentralized nature of dynamic teams leads to *curse of dimensionality* which impedes scalability. The partial observability requires minute analysis of the information structure involving *private* and *public* or *common* knowledge. The POMDP structure entails growing history of agents' observations and actions that leads to intractability. This calls for proper design of *belief* to circumvent such a growing history by leveraging Bayesian update, consequently requiring judicious choice of Bayesian inference to approximate the *posterior*. Moreover, in the temporal abstraction, the *option-policies* of the agents have stochastic termination, which adds to intricacies in the hierarchical reinforcement learning problem.

We study both planning and learning in the *team option-critic* framework. We propose *Distributed Option Critic*[1] (DOC) algorithm, where we leverage the notion of *common information approach* and *distributed policy gradient*. We employ the former to formulate a centralized (*coordinated*) system equivalent to the original decentralized system and to define the belief for the coordinated system. The latter is exploited in DOC for policy improvements of independent agents. We assume that there is a *fictitious coordinator* who observes the information shared by all agents, updates a belief on the joint-states in a Bayesian manner, chooses options and whispers them to the agents. The agents in turn use their private information to choose actions pertaining to the option assigned to it. Finally, the option-value of the cooperative game is learnt using distributed option-critic architecture.

Acknowledgements

We are indebted to Audrey Durand and Adriana Romero for their critical comments that helped us improve the content.

[1] We hereby would like to note that we have been unable to provide the proofs and some relevant preliminary ideas due to limitation of space; they are given in the full paper.

1 J-agent Dec-MDP planning with temporal abstraction

In this work we consider *goal-based-event-driven* Dec-MDP teams, where the agents know their own states but are unaware of others' states. There are some *goal* states $g \in \mathcal{G}$, where $\mathcal{G} \subset \mathcal{S}$ is a set of goal states, which the agents like to explore. There is a common pool of options $\mathcal{O}$ available to all agents, where we assume $|\mathcal{O}| =: J_{\text{option}} > J$. The agents choose an option o with unique identifier $i \in \{1, \ldots, J_{\text{option}}\}$ from the pool without replacement. We write $o^j = i$ to indicate that the option used by Agent j has the identifier i. The agents update their common belief of joint-states via *broadcasting*. Upon exploring a goal, the agents receive a common reward, they get penalized for collision and broadcasting involves a cost.

1.1 When to broadcast?

Any agent j at state s_t^j at time t broadcasts under two circumstances: if it reaches a goal state (i.e. $s_t^j \in \mathcal{G}$) and hence terminates its current option o^j deterministically, or greedily (1). Agent j can terminate its option o^j stochastically with probability $\beta^{o^j}(s_t^j)$, but does not necessarily broadcast. Let $\tilde{s}_t^j \coloneqq (\tilde{s}_t^1, \ldots, s_t^j, \ldots, \tilde{s}_t^J)$ be a joint-state sampled from the common belief (introduced in section 1.2), with the j-th component replaced by Agent j's true state s_t^j. At every step t, Agent j broadcasts if

$$Q^\mu(\tilde{s}_t^j, \mathbf{o}_{B_j}) > Q^\mu(\tilde{s}_t^j, \mathbf{o}_{\bar{B}_j}), \tag{1}$$

where Q^μ is the option-value corresponding to joint-option policy μ. The joint-option $\mathbf{o}_{B_j}$ implies that its component o^j is broadcast and joint-option $\mathbf{o}_{\bar{B}_j}$ implies its component o^j is not broadcast. While broadcasting, Agent j sends its state s_t^j, its action a_t^j and the ID of the option it was executing o^j to every other agents.

The aforementioned description tells us that broadcasting depends on the current state s_t^j of each Agent j and its current option o^j. For ease of exposition, let us denote the event that Agent j broadcasts at state s_t^j by $\text{Broad}(s_t^j, o^j)$. $\text{Broad}(s_t^j, o^j) = 1$ implies that Agent j has broadcast, and $\text{Broad}(s_t^j, o^j) = 0$ implies that it hasn't. The vector $\text{Broad}(\mathbf{s_t}, \mathbf{o}) \coloneqq (\text{Broad}(s_t^1, o^1), \ldots, \text{Broad}(s_t^J, o^J))$ denotes the broadcast symbols of all agents. More formally, at time t, $\text{Broad}(s_t^j, o^j)$ is given by:

$$\text{Broad}(s_t^j, o^j) \coloneqq \begin{cases} 1, & \text{if } s_t^j \in \mathcal{G} \text{ or (1) is true} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

1.2 Common information based belief state

The Dec-MDP can be viewed from the *common information* point-of-view as follows [4]. A *fictitious coordinator* observes the information available to all the agents, the *common information* and prescribes a *belief* on the joint-state and a *prescription* (Markov joint-option policy μ_t), which the agents apply in a decentralized manner (i.e., they apply their respective action-policies on their local states) to decide on their actions. The common information based belief (coordinator's state) on the joint-state $\mathbf{s}_t \in \mathcal{S}$ at time t is then defined as:

$$b_t^c(\mathbf{s}) \coloneqq \mathbb{P}(\mathbf{s}_t = \mathbf{s} \,|\, \mathcal{I}_t^{c,m}), \tag{3}$$

where $\mathcal{I}_t^{c,m}$ is the common information available to all agents at time instant m. Here m is the instant of the last broadcast.

Recall that when Agent j decides to broadcast, it broadcasts its own state s_t^j, and own action a_t^j. Hence, at any time t, the joint-observation $\mathbf{y}_t = (y_t^1, \ldots, y_t^J)$ made by all agents as follows:

$$y_t^j \coloneqq \begin{cases} (s_t^j, a_t^j), & \text{if } \text{Broad}(s_t^j, o_t^j) = 1 \\ \varnothing, & \text{otherwise.} \end{cases} \tag{4}$$

We can now express more formally the last instant of broadcast, m, as was introduced in $\mathcal{I}_t^{c,m}$ in (3) as $m \coloneqq \max\{m' \leq t : \text{for at least one } j, y_{m'}^j \neq \varnothing\}$. In other words, m is the last instant when at least one agent broadcast. Then, $\mathcal{I}_t^{c,m}$ can be defined as follows: $\mathcal{I}_t^{c,m} \coloneqq \{\mathbf{y}_{1:m}, \text{Broad}_{1:m}\}$, where $\text{Broad}_{1:m}$ is the history of broadcasting of all agents until time instant m. Note that since the broadcast information can be inferred from the joint-observation, $\text{Broad}_{1:t-1}$ can be absorbed in $y_{1:t-1}$ in the conditioning in the definition of b_t^c. Thus, (3) can be rewritten as $b_t^c(\mathbf{s}) \coloneqq \mathbb{P}(\mathbf{s}_t = \mathbf{s} \,|\, \mathbf{y}_{1:m})$. Clearly, $\mathbb{P}(\mathbf{s}_t = \mathbf{s} \,|\, \mathbf{y}_{1:t-1}) = \mathbb{P}(\mathbf{s}_t = \mathbf{s} \,|\, \mathbf{y}_{1:m})$, since no new information is received in times $\tau \in \{m+1, \ldots, t-1\}$. Thus,

$$b_t^c(\mathbf{s}) \coloneqq \mathbb{P}(\mathbf{s}_t = \mathbf{s} \,|\, \mathbf{y}_{1:t-1}). \tag{5}$$

From the common information perspective the coordinator observes the common information, the sequence of joint-observations until time t, $y_{1:t}$, and generates *prescriptions* (in our case this is the joint Markov option-policy) μ_t, according

to some *coordination rule* ψ such that $\psi : \Omega \to \mathcal{M}$, $j \in \mathcal{J}$ (where Ω is the set of all observations and $\mathcal{M}$ is the set of *Markov* option policies)

$$\mu_t = \psi(\mathbf{y}_{1:t-1}, \mu_{1:t-1}). \tag{6}$$

The prescription μ_t^j is then communicated to Agent j. Note that Agent j then uses the component μ_t^j of this prescription μ_t, chooses an option $o^j \sim \mu_t^j$, uses the action-policy π^{o^j} and termination probability β^{o^j} corresponding to o^j and generates its action a_t^j using its local information s_t^j as per $a_t^j \sim \pi^{o^j}(a_t^j|s_t^j)$. Since by (3), b_t^c is measurable with $(\mathbf{y}_{1:t-1}, \mu_{1:t-1})$, we can infer from (6) that there is no loss of optimality in restricting attention to coordination rules $\tilde{\psi}$ such that the prescription μ_t is given by $\mu_t = \tilde{\psi}(b_t^c)$.

Denote by $b_{t,t}^c$ the posterior based on current observation as given by $b_{t,t}^c(\mathbf{s}) \coloneqq \mathbb{P}(\mathbf{s}_t = \mathbf{s} \,|\, \mathbf{y}_{1:t})$.

Then, the evolution of the common belief is given by:

$$b_{t+1}^c(\mathbf{s}') = \mathbb{P}(\mathbf{s}_{t+1} = \mathbf{s}' \,|\, \mathbf{y}_{1:t}) = \sum_{\mathbf{s}\in\mathcal{S}} p^{\mathbf{a}}(\mathbf{s}, \mathbf{s}') b_{t,t}^c(\mathbf{s}), \tag{7}$$

where $p^{\mathbf{a}}(\mathbf{s}, \mathbf{s}')$ denotes the one-step transition probability of going from state $\mathbf{s}$ to state $\mathbf{s}'$ using action $\mathbf{a}$.

We show that the above described coordinated system is a POMDP with prescriptions μ_t and observations $\mathbf{y}_t = \tilde{h}_t(\mathbf{s}_t, \mu_t)$, where $\tilde{h}_t$ is Bayesian filtering update function[2]. Furthermore, based on a new joint observation received at time t, we show that the common information based belief b_t^c has a Bayesian update.

The optimal policy of the coordinated centralized system is the solution of a suitable dynamic program, i.e. the fixed point of which (if it exists) formulates the critic. We first show that the common information based belief state b_t^c is an *information state*, which forms a sufficient statistic to form a future belief b_{t+1}^c based on the current common belief and the current joint-option μ_t. Also, we establish the optimality of joint option-policy. We skip the main theorem involving these results due to lack of space and instead turn our focus to the learning problem, as described in the subsequent sections.

2 Learning in Dec-MDPs with options

In this section we consider a setup of J-agent cooperative game, where they have decentralized control policies but a single critic, the option-value, (infinite horizon discounted return) to maximize.

2.1 Common-belief based option-value and distributed gradient descent

We can extend the notion of option-value with full observability to the case with partial observability. Following the definition of *option-value upon arrival* with *call-and-return* option, we have the following:

$$U^{\mu}(b_t^c, \mathbf{o}) \coloneqq \sum_{s\in\mathcal{S}} U^{\mu}(\mathbf{s}, \mathbf{o}) b_t^c(\mathbf{s}) = \sum_{\mathbf{s}\in\mathcal{S}} \Big[\beta_{\text{none}}^{\mathbf{o}}(\mathbf{s}) Q^{\mu}(\mathbf{s}, \mathbf{o}) b_t^c(\mathbf{s}) + (1 - \beta_{\text{none}}^{\mathbf{o}}(\mathbf{s})) \max_{\mathcal{T}\in \text{Pow}(\mathcal{J})} \max_{\mathbf{o}'\in\mathcal{O}_{\text{avail}}(\mathcal{T})} Q^{\mu}(\mathbf{s}, \mathbf{o}') b_t^c(\mathbf{s})\Big], \tag{8}$$

where $\beta_{\text{none}}^{\mathbf{o}}(\mathbf{s}) \coloneqq \prod_{j\in\mathcal{J}}(1 - \beta^{o^j}(s^j))$ is the probability that no agent has terminated, $O_{\text{avail}}(\mathcal{T})$ is the set of available options for agents in the set $\mathcal{T} \subseteq \mathcal{J}$ and $\text{Pow}(\mathcal{J})$ is power-set of $\mathcal{J}$.

Q^{μ} in (8) is the solution of the following Bellman update:

$$Q^{\mu}(b_t^c, \mathbf{o}) \coloneqq \sum_{s\in\mathcal{S}} Q^{\mu}(\mathbf{s}, \mathbf{o}) b_t^c(\mathbf{s}) = \sum_{\mathbf{s}\in\mathcal{S}} \left(\sum_{\mathbf{a}\in\mathcal{A}} \boldsymbol{\pi}^{\mathbf{o}}(\mathbf{a}|\mathbf{s}) \Big[r^{\mathbf{a}}(\mathbf{s}) + \gamma \sum_{\mathbf{s}'\in\mathcal{S}} b_{t+1}^c(\mathbf{s}') \big(p^{\mathbf{a}}(\mathbf{s}, \mathbf{s}') U^{\mu}(\mathbf{s}', \mathbf{o})\big) \Big] \right) b_t^c(\mathbf{s}), \tag{9}$$

where $\boldsymbol{\pi}^{\mathbf{o}}(\mathbf{a}|\mathbf{s})$ is the joint probability of choosing joint-action $\mathbf{a}$ in joint-state $\mathbf{s}$. From independence of agents we have $\boldsymbol{\pi}^{\mathbf{o}}(\mathbf{a}|\mathbf{s}) \coloneqq \prod_{j\in\mathcal{J}} \pi^{o^j}(a^j|s^j)$, $r^{\mathbf{a}}(\mathbf{s})$ is the common immediate reward of choosing joint-action $\mathbf{a}$ in joint-state $\mathbf{s}$.

The optimal values corresponding to (8) and (9) are defined as follows:

$$U^*(b_t^c, \mathbf{o}) \coloneqq \max_{\mu\in\mathcal{M}} U^{\mu}(b_t^c, \mathbf{o}),\ Q^*(b_t^c, \mathbf{o}) \coloneqq \max_{\mu\in\mathcal{M}} Q^{\mu}(b_t^c, \mathbf{o}). \tag{10}$$

Define operator $\mathcal{B}$ as follows: $[\mathcal{B}Q^*](b_t^c, \mathbf{o}) \coloneqq \gamma \left(\sum_{\mathbf{a}\in\mathcal{A}} \boldsymbol{\pi}^{\mathbf{o}}(\mathbf{a}|\mathbf{s}) \sum_{\mathbf{s}'\in\mathcal{S}} b_{t+1}^c(\mathbf{s}') \big(p^{\mathbf{a}}(\mathbf{s}, \mathbf{s}') U^*(\mathbf{s}', \mathbf{o})\big) \right) b_t^c(\mathbf{s})$.

[2]Bayesian filtering applies Bayesian statistics and Baye's rule in solving Bayesian inference problems including stochastic filtering problems. See [3] and references therein for details.

Then, Q^* given by (10) can be rewritten as: with $r^{\mathbf{o}}$ is the immediate reward corresponding to joint-option $\mathbf{o}$, we have

$$Q^*(b_t^c, \mathbf{o}) = r^{\mathbf{o}}(b_t^c) + [\mathcal{B}Q^*](b_t^c, \mathbf{o}). \tag{11}$$

We show in the following lemma that $\mathcal{B}$ is a contraction. Thus, (11) has a unique solution. Furthermore, since $r^{\mathbf{o}}$ is bounded, so is Q^*.

In this work we assume that the agents are *factored, locally fully observable, transition independent and reward independent*. For such agents, it is shown in [5] that *distributed gradient descent* achieves local optima. In the sequel, we propose the idea of learning in Dec-POMDP by augmenting the notion of distributed gradient descent of [5] with the spirit of option-critic algorithm [2]. For Agent j we consider the parameters $\boldsymbol{\theta}^j = (\theta^{j_1}, \ldots, \theta^{j_M})$ for action-policy π^{o^j} and $\boldsymbol{\phi}^j = (\phi^{j_1}, \ldots, \phi^{j_M})$ for the termination function β^{o^j}, where $M \geq 1$ is the number of parameters. We write $\pi^{o^j, \boldsymbol{\theta}^j}$ and $\beta^{o^j, \boldsymbol{\phi}^j}$ to show the parameterized action-policy and termination function. We assume the standard Boltzmann function for the action policy as given by

$$\pi^{o^j, \boldsymbol{\theta}^j}(a^j \mid s^j) = \frac{e^{Q^{o^j}_{\text{intra}}(s^j, a^j)}}{\sum_{a^j \in \mathcal{A}^j} e^{Q^{o^j}_{\text{intra}}(s^j, a^j)}}, \tag{12}$$

where $Q^{o^j}_{\text{intra}}(s^j, a^j)$ is the weight of the Boltzmann intra-option policy $\pi^{o^j, \boldsymbol{\theta}^j}$ of Agent j. Given the option o^j prescribed by the coordinator, each agent j maintains its own action-value $Q^{o^j}_{\text{intra}}(s^j, a^j)$ for their own true state s^j, and action $a^j \in \mathcal{A}^j$. With fixed o^j, the weights $Q^{o^j}_{\text{intra}}(s^j, a^j)$ are updated using vanilla Q-learning.

2.2 Factored common belief

For large scale systems, the common belief is intractable due to the combinatorial nature of joint state-space. Thus in one of our experiments we assume that the common belief is *factored*, i.e.,

$$b_t^c(\mathbf{s}) \coloneqq \mathbb{P}(\mathbf{s}_t = \mathbf{s} \,|\, \mathbf{y}_{1:t-1}) \approx \prod_{j \in \mathcal{J}} \mathbb{P}(s_t^j = s^j \,|\, \mathbf{y}_{1:t-1}) =: \prod_{j \in \mathcal{J}} b_t^{c,j}(s^j) =: b_t^{c,\text{fact}}(\mathbf{s}). \tag{13}$$

2.2.1 Update of factored common belief in a consistent way

When the coordinator updates the factored common belief based on the joint-observation $\mathbf{y}_t$, it can do so first by iteratively updating the factored likelihood L_t as given by the following: for each agent $j \in \mathcal{J}$

$$\begin{aligned} L_t(s^j) &\coloneqq \mathbb{P}(\text{br}^j, a^j | s^j, \text{Broad}, b_t^{c,j}) \\ &\propto L_{t-1}(s^j) \frac{\mathbb{E}_{s_t^j \sim b_t^{c,j}} \Big(\mathbb{1}(s_t^j = s^j) \mathbb{1}(\text{Broad}(s^j) = \text{br}^j) \mathbb{1}(\pi^{o^j}(s^j) = a^j) \Big)}{\mathbb{E}_{s_t^j \sim b_t^{c,j}} \mathbb{1}(s_t^j = s^j)}, \end{aligned} \tag{14}$$

where Broad is the deterministic broadcast function and $\text{br}^j \coloneqq \text{Broad}(s^j)$, is the broadcast symbol of agent j, $\text{br}^j \in \{0, 1\}$.

Then, the posterior of the factored common belief can be computed using Bayes update rule (**??**) using the factored likelihood (14). But this posterior may not be consistent with the fact that the dynamics of the agents are not transition independent (e.g., when collision is not allowed as in our case). So, in order to make the posterior *consistent*, the coordinator observes the current joint observation $\mathbf{y}_t$ and re-marginalizes the factored common belief as follows: for inner loop k, do for all $j \in \mathcal{J}$

$$b^{0,j} = b_t^{c,j}, \quad b^0 = \prod_{j \in \mathcal{J}} b^{0,j} \tag{15}$$

$$\begin{aligned} b^{k+1,j} &\propto \mathbb{E}_{\mathbf{s} \sim b^k} L_t(s^j) \mathbb{P}(s^j \neq \mathbf{s}^{-j}) \\ &\propto \mathbb{E}_{\mathbf{s} \sim b^k} L_t(s^j) \Big[\mathbb{1}(\mathbf{y}_t^{-j} \neq \text{None}) \mathbb{P}(\mathbf{y}_t^{-j} \neq s^j)] + \mathbb{1}(\mathbf{y}_t^{-j} = \text{None}) \sum_{\mathbf{s}^{-j} \sim b^{k,-j}} \mathbb{P}(\mathbf{s}^{-j} \neq s^j) \Big], \end{aligned} \tag{16}$$

where for any agent j, $\mathbf{y}_t^{-j}$, $\mathbf{s}^{-j}$ and $b^{k,-j}$ denote respectively the observations, sampled states and the factored beliefs of other agents. With a slight abuse of notation, the equalities and inequalities in the last line imply for all other agents $k \in \mathcal{J} \setminus j$.

Using arguments for the convergence of the policy-gradient based algorithms (e.g., [7]) and the local optima achieved by distributed stochastic gradient descent [5, Theorem 1], we can show the following

Theorem 1 *The DOC algorithm given above converges to the optimal option-value Q^*.*

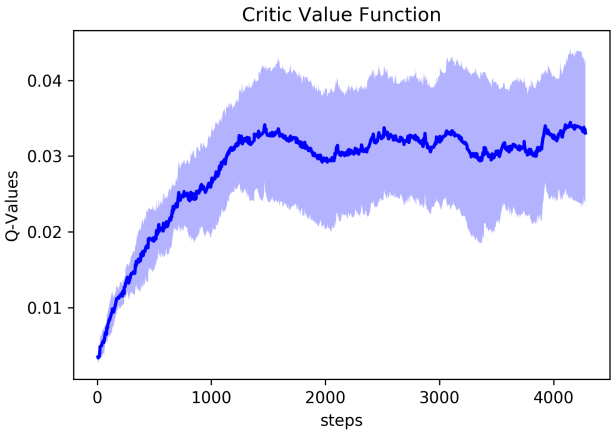


Figure 1: Option-critic values over iterations

3 Experiments

We validate theoretical results using *Four-room* environment introduced in [6]. Unlike their setup, we do not have any pre-specified options in the common pool of options available to all agents. In the first experiment we investigate a navigation task in a tabular setting with several equivalent targets. In the second experiment we use a *Teamgrid* [1] environment with hierarchichal tasks which call for cooperation among the agents. We parameterize the Q-value, intra-option policy, termination function and the broadcast function with deep neural nets and use factored common belief so that the results are scalable to large state-spaces. Fig. 1 shows the convergence over 5 runs of option-critic for the tabular set-up.

4 Discussion

This paper investigates the temporal abstraction in dynamic teams (cooperative multi-agent systems) with expensive broadcast. The main contribution of the paper is the establishment of the theoretical results of the main learning algorithm. We extend the option-critic architecture to multi-agent systems and provide convergence results. In the underlying planning problem, we adopt the common-information approach which transforms the decentralized problem into an equivalent centralized set-up which enables us to leverage the tools from centralized stochastic optimization problems. Note that similar to option-critic, DOC does not require learning the the options and learning the intra-option policies and terminations suffices. For simplicity of exposition we have assumed in this work that whenever the agents decide to broadcast, they broadcast their state and action. This is common in practice in applications such as in communication networks with collision channels where the alphabet to be transmitted is of a few bits and the action of *transmitting or not* is of one bit. For larger bits to be broadcast, one may use techniques like *source-coding* before transmitting, but that would not alter the rest of the modeling assumptions and the convergence results will continue to hold. In such applications collision in the channel causes the packet to drop and so putting collision penalty to discourage collision helps in learning. That the agents broadcast to everyone can be costly in realistic scenarios. In that case one may learn optimal neighbourhood on top of learning the policies and terminations and broadcast only to the neighbours.

References

[1] Teamgrid github repo. `https://github.com/mila-iqia/teamgrid`.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2017.

[3] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69, Jan. 2003.

[4] A. Nayyar, A. Mahajan, and D. Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. 58(7):1644–1658, jul 2013.

[5] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, pages 489–496, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[6] Richard Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[7] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

Improving Generalization over Large Action Sets

Yash Chandak
University of Massachusetts Amherst
ychandak@cs.umass.edu

Georgios Theocharous
Adobe Research, San Jose
theochar@adobe.com

James E. Kostas
University of Massachusetts Amherst
jekostas@cs.umass.edu

Scott M. Jordan
University of Massachusetts Amherst
sjordan@cs.umass.edu

Philip S. Thomas
University of Massachusetts Amherst
pthomas@cs.umass.edu

Abstract

Most model-free reinforcement learning methods leverage state representations (embeddings) for generalization, but either ignore structure in the space of actions or assume the structure is provided *a priori*. We show how a policy can be decomposed into a component that acts in a low-dimensional space of action representations and a component that transforms these representations into actual actions. These representations improve generalization over large, finite action sets by allowing the agent to infer the outcomes of actions similar to actions already taken. We provide an algorithm to both learn and use action representations and provide conditions for its convergence. The efficacy of the proposed method is demonstrated on large-scale real-world problems.

Keywords: Large Action Spaces, Action Representations

Acknowledgements

Part of the work was done when YC was an intern at Adobe. Later, the work was supported by Adobe Research Grant.

1 Introduction

Reinforcement learning (RL) methods have been applied successfully to many simple and game-based tasks. However, their applicability is still limited for problems involving decision making in many real-world settings. One reason is that many real-world problems with significant human impact involve selecting a single decision from a multitude of possible choices. For example, maximizing long-term portfolio value in finance using various trading strategies [4], improving fault tolerance by regulating voltage level of all the units in a large power system [3], and personalized tutoring systems for recommending sequences of videos from a large collection of tutorials. Therefore, it is important that we develop RL algorithms that are effective for real problems, where the number of possible choices is large.

In this paper we consider the problem of creating RL algorithms that are effective for problems with large action sets. Existing RL algorithms handle large *state* sets (e.g., images consisting of pixels) by learning a representation or embedding for states (e.g., using line detectors or convolutional layers in neural networks), which allow the agent to reason and learn using the state representation rather than the raw state. We extend this idea to the set of actions: we propose learning a representation for the actions, which allows the agent to reason and learn by making decisions in the space of action representations rather than the original large set of possible actions. This setup is depicted in Figure 1, where an *internal policy*, π_i, acts in a space of action representations, and a function, f, transforms these representations into actual actions. Together we refer to π_i and f as the *overall policy*, π_o.

Recent work has shown the benefits associated with using action-embeddings [2], particularly that they allow for generalization over actions. For real-world problems where there are thousands of possible (discrete) actions, this generalization can significantly speed learning. However, this prior work assumes that fixed and predefined representations are provided. In this paper we present a method to autonomously learn the underlying structure of the action set by using the observed transitions. This method can both learn from scratch and improve upon a provided action representation.

A key component of our proposed method is that it frames the problem of learning an action representation (learning f) as a *supervised* learning problem rather than an RL problem. This is desirable because supervised learning methods tend to learn more quickly and reliably than RL algorithms since they have access to instructive feedback rather than evaluative feedback [6]. The proposed learning procedure exploits the structure in the action set by aligning actions based on the similarity of their impact on the state. Therefore, updates to a policy that acts in the space of learned action representation generalizes the feedback received after taking an action to other actions that have similar representations.

To evaluate our proposed method empirically, we study two real-world recommender system problems using data from Adobe HelpX and Adobe Photoshop. In both the applications, there are thousands of possible recommendations that could be given at each time step (e.g., which video to suggest the user watch next on the HelpX portal, or which tool to suggest to the user next in the Photoshop software). Our experimental results show our proposed system's ability to significantly improve performance relative to existing methods for these applications by quickly and reliably learning action representations that allow for meaningful generalization over the large discrete set of possible actions.

2 Background

We consider problems modeled as discrete-time *Markov decision processes* (MDPs) with discrete states and finite actions. An MDP is represented by a tuple, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, d_0)$. $\mathcal{S}$ is the set of all possible states, called the state space, and $\mathcal{A}$ is a finite set of actions, called the action set. In this work, we restrict our focus to MDPs with finite action sets, and $|\mathcal{A}|$ denotes the size of the action set. The random variables, $S_t \in \mathcal{S}$, $A_t \in \mathcal{A}$, and $R_t \in \mathbb{R}$ denote the state, action, and reward at time $t \in \{0, 1, \dots\}$. The first state, S_0, comes from an initial distribution, d_0, and the reward function $\mathcal{R}$ is defined so that $\mathcal{R}(s, a) = \mathbf{E}[R_t | S_t = s, A_t = a]$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The reward discounting parameter is given by $\gamma \in [0, 1)$. $\mathcal{P}$ is the state transition function. A policy $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a conditional distribution over actions for each state. For any policy π, the corresponding state-action value function, $q^\pi(s, a)$, and the state value function, $v^\pi(s)$, are defined as in [6].

3 Generalization over Actions

The benefits of capturing the structure in the underlying state space of MDPs is a well understood and a widely used concept in RL. State representations allow the policy to generalize across states. Similarly, there often exists additional structure in the space of actions that can be leveraged. We hypothesize that exploiting this structure can enable quick generalization across actions, thereby making learning with large action sets feasible. To bridge the gap, we introduce an action representation space, $\mathcal{E} \subseteq \mathbb{R}^d$, and consider a factorized policy, π_o, parameterized by an embedding-to-action mapping function, $f \colon \mathcal{E} \to \mathcal{A}$, and an internal policy, $\pi_i \colon \mathcal{S} \times \mathcal{E} \to [0, 1]$, such that the distribution of A_t given S_t is characterized by:

$$E_t \sim \pi_i(\cdot | S_t), \qquad A_t = f(E_t).$$

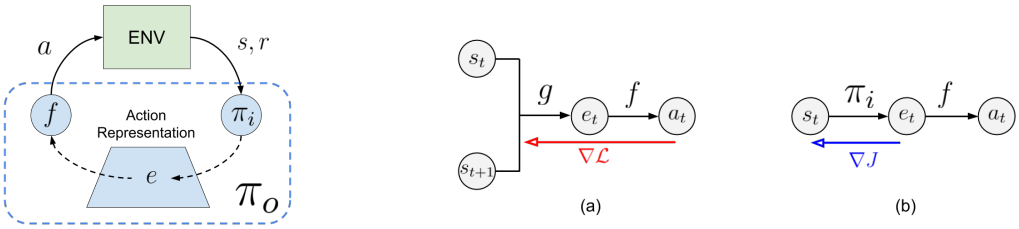


Figure 1: (Left) The structure of the proposed overall policy, π_o, consisting of f and π_i, that learns action representations to generalize over large action sets. (Right) a) Given a state transition tuple, functions g and f are used to estimate the action taken. The red arrow denotes the gradients of the supervised loss (2) for learning the parameters of these functions. b) During execution, an internal policy, π_i, can be used to first select an action representation, e. The function f, obtained from previous learning procedure, then transforms this representation to an action. The blue arrow represents the internal policy gradients (3) obtained using Lemma 1 to update π_i.

Here, π_i is used to sample $E_t \in \mathcal{E}$, and the function f deterministically maps this representation to an action in the set $\mathcal{A}$. Both these components together form an *overall policy*, π_o. With a slight abuse of notation, we use $f^{-1}(a)$ to denote the set of representations that are mapped to the action a by the function f, i.e., $f^{-1}(a) \coloneqq \{e \in \mathcal{E} : f(e) = a\}$. With this, we define the overall policy, $\pi_o(a|s) \coloneqq \int_{f^{-1}(a)} \pi_i(e|s)\,\mathrm{d}e$. In the following sections, we present the supervised learning process for the function f when π_i is fixed. Next we give the policy gradient learning process for π_i when f is fixed. Finally, we combine these methods to learn f and π_i simultaneously.

Supervised Learning of f For a Fixed π_i : We leverage a standard Markov property, often used for learning probabilistic graphical models, to express $P(A_t|S_t, S_{t+1})$ as $\int_{\mathcal{E}} P(A_t|E_t = e)P(E_t = e|S_t, S_{t+1})\,\mathrm{d}e$. Given an embedding E_t we assume that the action, A_t, is deterministic and can be represented by a function $f : \mathcal{E} \to \mathcal{A}$, such that $P(A_t|S_t, S_{t+1})$ can be decomposed in terms of f and $P(E_t|S_t, S_{t+1})$. However, such a function f that maps from representation space to the actions may not be known *a priori*. We propose searching for an estimator, $\hat{f}$, of f and an estimator, $\hat{g}(E_t|S_t, S_{t+1})$, of $P(E_t|S_t, S_{t+1})$ so that a reconstruction of $P(A_t|S_t, S_{t+1})$ is accurate. Let this estimate of $P(A_t|S_t, S_{t+1})$ based on $\hat{f}$ and $\hat{g}$ be $\hat{P}(A_t|S_t, S_{t+1}) = \int_{\mathcal{E}} \hat{f}(A_t|E_t = e)\hat{g}(E_t = e|S_t, S_{t+1})\,\mathrm{d}e$. One way to measure the difference between $P(A_t|S_t, S_{t+1})$ and $\hat{P}(A_t|S_t, S_{t+1})$ is using the expected (over states coming from the on-policy distribution) Kullback-Leibler (KL) divergence

$$= -\,\mathbf{E}\left[\sum_{a\in\mathcal{A}} P(a|S_t, S_{t+1}) \ln\left(\frac{\hat{P}(a|S_t, S_{t+1})}{P(a|S_t, S_{t+1})}\right)\right] \tag{1}$$

Since the observed transition tuples, (S_t, A_t, S_{t+1}), contain the action responsible for the given S_t to S_{t+1} transition, an on-policy sample estimate of the KL-divergence can be computed readily using (1). We adopt the following loss function based on the KL divergence between $P(A_t|S_t, S_{t+1})$ and $\hat{P}(A_t|S_t, S_{t+1})$:

$$\mathcal{L}(\hat{f}, \hat{g}) = -\mathbf{E}\left[\ln\left(\hat{P}(A_t|S_t, S_{t+1})\right)\right], \tag{2}$$

where the denominator in (1) is not included in (2) because it does not depend on $\hat{f}$ or $\hat{g}$. If $\hat{f}$ and $\hat{g}$ are parameterized, their parameters can be learned by minimizing the loss function, $\mathcal{L}$, using a supervised learning procedure. In our experiments, f contains learnable representations for the actions, and maps an embedding to the closest action. A computational graph for this model is shown in Figure 1. Note that, while $\hat{f}$ will be used for f in an overall policy, $\hat{g}$ is only used to find $\hat{f}$, and will not serve an additional purpose. As this supervised learning process only requires estimating $P(A_t|S_t, S_{t+1})$, it does not require (or depend on) the rewards. This partially mitigates the problems due to sparse and stochastic rewards, since an alternative informative supervised signal is always available. This is advantageous for making the action representation component of the overall policy learn quickly and with low variance updates.

Learning π_i For a Fixed f : A common method for learning a policy parameterized with weights θ is to optimize the discounted start-state objective function, $J(\theta) \coloneqq \sum_{s\in\mathcal{S}} d_0(s)v^{\pi}(s)$. For a policy with weights θ, the expected performance of the policy can be improved by ascending the *policy gradient*, $\frac{\partial J(\theta)}{\partial\theta}$. Let the state-value function associated with the internal policy, π_i, be $v^{\pi_i}(s) = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_t|s, \pi_i, f]$, and the state-action value function $q^{\pi_i}(s, e) = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_t|s, e, \pi_i, f]$. We then define the performance function for π_i as, $J_i(\theta) \coloneqq \sum_{s\in\mathcal{S}} d_0(s)v^{\pi_i}(s)$. Viewing the embeddings as the action for the agent with policy π_i, the policy gradient theorem [7], states that the gradient of $J_i(\theta)$ is,

$$\frac{\partial J_i(\theta)}{\partial\theta} = \sum_{t=0}^{\infty} \mathbf{E}\Big[\gamma^t \int_{\mathcal{E}} q^{\pi_i}(S_t, e)\frac{\partial}{\partial\theta}\pi_i(e|S_t)\,\mathrm{d}e\Big], \tag{3}$$

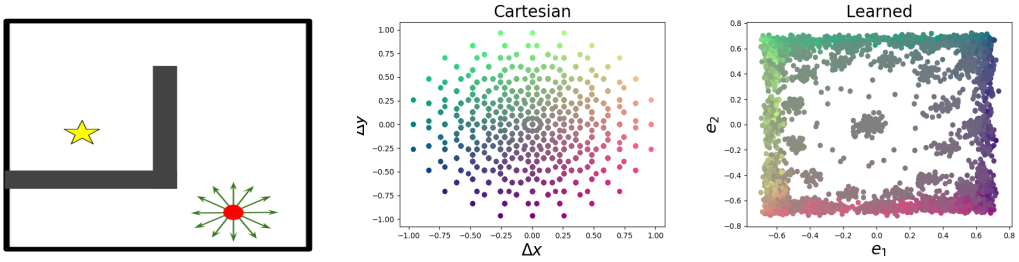


Figure 2: (a) The maze environment. The star denotes the goal state, the red dot corresponds to the agent and the arrows around it are the 12 actuators. (b) 2-D representations for the displacements in the Cartesian co-ordinates caused by each action, and (c) learned action embeddings. In both (b) and (c), each action is colored based on the displacement (Δx, Δy) it produces. Cartesian actions are plotted on co-ordinates (Δx, Δy), and learned ones are on the coordinates in the embedding space. Smoother color transition corresponds to preservation of the *relative* underlying structure.

where, the expectation is over states from d^π, as defined by [7] (which is not a true distribution, since it is not normalized). The parameters of the internal policy can be learned by iteratively updating its parameters in the direction of $\partial J_i(\theta)/\partial\theta$. Since there are no special constraints on the policy π_i, any policy gradient algorithm designed for continuous control can be used out-of-the-box.

However, note that the performance function associated with the overall policy, π_o (consisting of function f and the internal policy parameterized with weights θ), is, $J_o(\theta, f) = \sum_{s\in\mathcal{S}} d_0(s) v^{\pi_o}(s)$. The ultimate requirement is the improvement of this overall performance function, $J_o(\theta, f)$, and not just $J_i(\theta)$. So, how useful is it to update the internal policy, π_i, by following the gradient of its own performance function? The following lemma answers this question.

Lemma 1. *For all deterministic functions, f, which map each point, $e \in \mathbb{R}^d$, in the representation space to an action, $a \in \mathcal{A}$, the expected updates to θ based on $\frac{\partial J_i(\theta)}{\partial\theta}$ are equivalent to updates based on $\frac{\partial J_o(\theta,f)}{\partial\theta}$. That is, $\frac{\partial J_o(\theta,f)}{\partial\theta} = \frac{\partial J_i(\theta)}{\partial\theta}$.*

The chosen parameterization for the policy has this special property, which allows π_i to be learned using its internal policy gradient. Since this gradient update does not require computing the value of any $\pi_o(a|s)$ explicitly, the potentially intractable computation of f^{-1} required for π_o can be avoided. Instead, $\partial J_i(\theta)/\partial\theta$ can be used directly to update the parameters of the internal policy while still optimizing the overall policy's performance, $J_o(\theta, f)$.

Learning π_i and f Simultaneously: Since the supervised learning procedure for f does not require rewards, a few initial trajectories can contain enough information to begin learning a useful action representation. As more data becomes available it can be used for fine-tuning and improving the action representations.

If the action representations are held fixed while learning the internal policy, then as a consequence of Lemma 1, convergence of our algorithm directly follows from previous two-timescale results [1]. Learning both π_i and f simultaneously using our PG-RA algorithm can also be shown to converge. We consider three learning rate sequences, such that the update recursion for the internal policy is on the slowest timescale, the critic's update recursion is on the fastest, and the action representation module's has an intermediate rate. With this construction, we leverage the three-timescale analysis technique [1] for convergence. Formal proofs and exact implementation details are left out due to space constraints.

Algorithm 1: Policy Gradient with Representations for Action (PG-RA)

```
Initialize action representations
for episode = 0, 1, 2... do
    for t = 0, 1, 2... do
        Sample action embedding, E_t, from π_i(·|S_t)
        A_t = f̂(E_t)
        Execute A_t and observe S_{t+1}, R_t
        Update π_i using any policy gradient algorithm
        Update critic (if any) to minimize TD error
        Update f̂ and ĝ to minimize ℒ defined in (2)
```

4 Empirical Analysis

A core motivation of this work is to provide an algorithm that can be used as a drop-in extension for improving the action generalization capabilities of existing policy gradient methods for problems with large action spaces. We consider two standard policy gradient methods: actor-critic (AC) and deterministic-policy-gradient (DPG) in our experiments.

Maze: As a proof-of-concept, we constructed a continuous-state maze environment where the state comprised of the coordinates of the agent's current location. The agent has n equally spaced actuators (each actuator moves the agent in the direction the actuator is pointing towards) around it, and it can choose whether each actuator should be on or off.

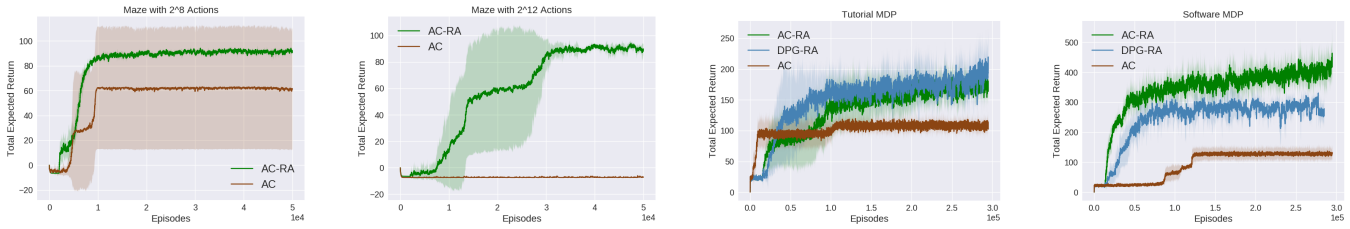


Figure 3: Results for the Maze domain with 2^8 actions, 2^{12} actions, Adobe HelpX MDP and Adobe Photoshop MDP. AC-RA (green) and DPG-RA (blue) are the variants of PG-RA algorithm that uses actor-critic (red) and DPG, respectively.

Therefore, the size of the action set is exponential in the number of actuators, that is $|\mathcal{A}| = 2^n$. The net outcome of an action is the vectorial summation of the displacements associated with the selected actuators. The agent is rewarded with a small penalty for each time step, and a reward of 100 is given upon reaching the goal position. To make the problem more challenging, random noise was added to the action 10% of the time and the maximum episode length was 150 steps. This environment is a useful test bed as it requires solving a long horizon task in an MDP with a large action set and a single goal reward. The visualizations of the learned action representations on the maze domain is provided in Figure 2.

Real-word recommender systems: We consider two real-world applications of recommender systems that require decision making over *multiple time steps*. First, Adobe HelpX, a web-based video-tutorial platform, which has a recommendation engine that suggests a series of tutorial videos on various Adobe software products. The second application is Adobe Photoshop, a professional multi-media editing software. The aim is to meaningfully engage the users in learning how to use these software products and convert novice users into experts in their respective areas of interest. For both of these applications, an existing log of user's click stream data was used to create an n-gram based MDP model for user behavior [5]. Sequences of user interaction were aggregated to obtain over 29 million clicks and 1.75 billion user clicks for HelpX and Photoshop, respectively. 1498 tutorials and 1843 tools for the HelpX platform and Adobe Photoshop, respectively, were used to create the action set for the MDP model. Rewards were chosen based on a surrogate measure for difficulty level of tutorials on HelpX portal and popularity of final outcomes of user interactions in Photoshop, respectively.

Performance Improvement: The plots in Figure 3 for the Maze domain show how the performance of standard actor-critic (AC) method deteriorates as the number of actions increases, even though the goal remains the same. However, with the addition of an action representation module it is able to capture the underlying structure in the action space and consistently perform well across all settings. Similarly, for both Adobe HelpX and Adobe Photoshop MDPs, standard AC methods fail to reason over longer time horizons under such an overwhelming number of actions, choosing mostly one-step actions that have high returns. In comparison, instances of our proposed algorithm are not only able to achieve significantly higher return, up to $2\times$ and $3\times$ in the respective tasks, but they do so much quicker. These results reinforce our claim that learning action representations allow implicit generalization of feedback to other actions embedded in proximity to executed action.

Further, under the PG-RA algorithm, only a fraction of total parameters, the ones in the internal policy, are learned using the high variance policy gradient updates. The other set of parameters associated with action representations are learned by a supervised learning procedure. As evident from the plots in the Figure 3, this reduces the variance of updates significantly, thereby making the PG-RA algorithms learn a better policy faster. These advantages allow the internal policy, π_i, to quickly approximate an optimal policy without succumbing to the curse of large actions sets.

References

[1] V. S. Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

[2] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[3] M. Glavic, R. Fonteneau, and D. Ernst. Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine*, 50(1):6918–6927, 2017.

[4] Z. Jiang, D. Xu, and J. Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

[5] G. Shani, D. Heckerman, and R. I. Brafman. An MDP-based recommender system. *Journal of Machine Learning Research*, 2005.

[6] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[7] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Discrete off-policy policy gradient using continuous relaxations

Andre Cianflone
Mila - McGill University

Zafarali Ahmed
Mila - McGill University

Riashat Islam
Mila - McGill University

Avishek Joey Bose
Mila - McGill University

William L. Hamilton
Mila - McGill University

Abstract

Off-Policy policy gradient algorithms are often preferred to on-policy algorithms due to their sample efficiency. Although sound off-policy algorithms derived from the policy gradient theorem exist for both discrete and continuous actions, their success in discrete action environments have been limited due to issues arising from off-policy corrections such as importance sampling. This work takes a step in consolidating discrete and continuous off-policy methods by adapting a low-bias, low-variance continuous control method by relaxing a discrete policy into a continuous one. This relaxation allows the action-value function to be differentiable with respect to the discrete policy parameters, and avoids the importance sampling correction typical of off-policy algorithms. Furthermore, the algorithm automatically controls the amount of relaxation, which results in implicit control over exploration. We show that the relaxed algorithm performs comparably to other off-policy algorithms with less hyperparameter tuning.

Keywords: policy gradient, off-policy actor-critic, continuous relaxation

Acknowledgements

Z.A. is funded by a Canada Graduate Scholarship, A.C. is funded by a Borealis AI Fellowship.

1 Introduction

Policy gradient methods are a class of algorithms used to solve reinforcement learning problems (RL) by directly optimizing a parameterized policy. *On-policy learning* uses data collected from this policy to compute gradient updates. Despite being rather successful [1], they can be sample inefficient as new data needs to be collected for each gradient update. Consequently, *Off-policy learning* is preferred due to its ability to re-use data collected from older policies. In particular, off-policy methods support data re-use from multiple behaviour policies, while learning a desired target policy.

While algorithms such as the Deep Deterministic Policy Gradient (Deep DPG) [2] exist for environments with continuous actions, there has not been much progress for discrete actions due to the lack of a viable discrete reparameterization approach. Algorithms like off-policy actor critic (Off-PAC) [3] and Actor Critic with Experience Replay (ACER) [4] can be derived for discrete action environments. However, the reliance on the importance sampling corrections limits their use in practice due to its high variance gradient estimate [5]. Recent work introduces Actor-Critic with Emphatic Weightings (ACE) [6] as another approach to discrete action off-policy learning that introduces the first "off-policy policy gradient theorem". However, ACE also requires estimating corrections and has not yet been demonstrated in more complex domains.

Our work aims to use successful continuous control algorithms [7] for discrete action environments by using *continuous relaxations* of samples from a discrete policy [8]. In essence, we convert the learning of a discrete policy into a continuous control problem. A particularly interesting side effect of the relaxation is the introduction of a temperature parameter, τ, that controls the amount of relaxation: The temperature can be automatically tuned [9], thereby controlling the entropy of the policy and eliminating the need for external exploration noise. We call this approach a Autotuned, Relaxed, Reparameterized Discrete Domain algorithm (AR2D2). Our contributions are:

1. Using continuous relaxations of discrete categorical samples [10, 8] to find the gradient of the action-value function, resulting in an algorithm similar to DPG [11].
2. Automatic control of the relaxation allowing sufficient exploration and eventual recovery of the optimal policy using a novel objective that balances variance reduction [9] and action-value maximization.

2 Background

We start by covering the off-policy reinforcement learning setting before considering the continuous relaxation in Section 2.1. Consider a Markov decision process $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of discrete actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state-transition probabilities, and $\gamma \in [0, 1]$ is the discount factor. The expected discounted return from a state, s_0, is given by the value function: $V^\pi(s) = \mathbb{E}_\pi[\sum_t \gamma^t r_t | s_0 = s]$. In policy gradient methods, we search for a parameterized *target* policy, π_θ, that maximizes $J(\theta) = \sum_s d_{\pi_\theta}(s) V^{\pi_\theta}(s)$ where $d_{\pi_\theta}(s)$ is the stationary state distribution under the policy π_θ.

However, in case of off-policy learning the samples are drawn from the state distribution under the *behaviour policy*, $\mu(s|a)$. Therefore, we optimize $J(\theta) = \sum_s d_\mu(s) V^{\pi_\theta}(s)$ where $d_\mu(s)$ is the stationary state distribution under μ.

Importance sampling techniques (IS) can be used to correct for the discrepancy in the behaviour and target policies [5]. However, IS corrections, being high variance, often make algorithms such as Off-PAC [3] difficult to use in practice. Alternatively, we can use the deterministic policy gradient theorem [12] to avoid IS corrections by considering deterministic policies, $\pi_\theta(a|s) = a$. In particular, DPG proposes a variant of Q-learning for policy gradients, where instead of taking a greedy policy improvement, we can directly improve the policy in the direction of the gradient of the action-value function: $\nabla_\theta J(\theta) = \sum_s d_\mu(s) \nabla_\theta Q^{\pi_\theta}(s, \pi_\theta(s))$. One limitation of the DPG is that it requires differentiable samples.

Differentiable reparameterizations exist for continuous distributions like the Gaussian [13, 14] and have been applied to continuous control problems in RL [11, 7]. While relaxing a categorical distribution has been explored in reinforcement learning as a action-depentent control variate [9] and a policy [15], it has not been fully developed into a viable alternative to well-known algorithms such as DQN [16].

2.1 Continuous Relaxations for Discrete Variables

In this section, we cover background material related to discrete reparmetrization of categorical distributions [10, 8]. Consider the general objective of optimizing parameters θ of a probability distribution, p_θ, to maximize the function f. The gradient is defined as $\nabla_\theta L(\theta) = \nabla_\theta \, \mathbb{E}_{z \sim p_\theta}[f(z)]$. When f is not differentiable the log-deriviative identity[1] can be applied to obtain the REINFORCE estimator, $\nabla_\theta L(\theta) = \mathbb{E}_{z \sim p_\theta(z)}[f(z) \nabla_\theta \log p_\theta(z)]$ which can be estimated using Monte-Carlo sampling [17].

[1]The log-derivative identity is $\nabla x = x \nabla \log x$

In cases where f is differentiable and p_θ can be *reparameterized* through a deterministic function, $z = g(\epsilon, \theta)$, a low variance gradient estimate can be be computed by shifting the stochasticity from the distributional parameters to a standardized noise model, ϵ [13, 14]. Specifically, we can rewrite the gradient computation as $\nabla_\theta L(\theta) = \mathbb{E}_\epsilon[\nabla_g f(g(\epsilon, \theta))\nabla_\theta g(\epsilon, \theta)]$.

The Gumbel-Max trick [18] offers such a reparameterization for the categorical distribution: $z = \arg\max_i[g_i + \log \eta_i]$ where η_i are the log probabilities for a Categorical distribution and g_i are independent and identically distributed noise variables from the Gumbel$(0, 1)$ distribution. While such a reparametrization shifts the distribution parameters to a deterministic node, it introduces a non-differentiable $\arg\max$. The Gumbel-Softmax (GS) [10] distribution proposes to replace the $\arg\max$ with a softmax and temperature parameter τ:

$$y_i = \frac{\exp((\log \eta_i + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log \eta_j + g_j)/\tau)} \tag{1}$$

where $\tau \to 0$ recovers the $\arg\max$, and $\tau \to \infty$ recovers the uniform distribution. Due to the relaxation, the softmax operation is differentiable providing continuous differentiable samples from this distribution. Computing $\arg\max_i y_i$ corresponds to sampling from a categorical distribution and allows execution in a reinforcement learning environment.

3 Off-Policy Policy Gradients with Gumbel Reparameterization

In this section we discuss how to introduce the Gumbel-Softmax as an alternate parameterized policy for discrete actions in the off-policy setting. We will do this by deriving the gradient for the action-value function to do policy improvement by following the grading direction.

Recall the off-policy learning setup, where the goal is to learn a target parameterized policy π_θ while collecting data from a behaviour policy μ. Consider the gradient of the action-value function:

$$\nabla_\theta J(\pi_\theta) = E_{s\sim d_\mu(s)}[\nabla_a Q(s, a)\nabla_\theta \pi_\theta(s)] \tag{2}$$

Like in DPG, $a = \pi_\theta(s)$, where π_θ is implemented using a Gumbel-Softmax policy with a relaxation parameter, τ (Equation 1). These relaxed discrete actions allow us to take gradients of Q w.r.t. the policy parameters θ, effectively back-propagating through the sampling process. To execute actions in the environment, continuous samples from relaxed policies are discretized using $\arg\max$ so that they correspond to samples from a categorical distribution. In supervised learning tasks, the Gumbel-Softmax temperature parameter is decayed [10] to reduce the relaxation over time. In reinforcement learning, premature annealing may lead to a suboptimal deterministic policy as the policy would fail to sample a diverse number of trajectories (i.e. reduced exploration). The temperature, τ, must be carefully controlled to prevent this outcome. In this work we consider τ is learned during optimization.

We now describe an off-policy actor-critic algorithm we call AR2D2 . We first describe the standard critic update to learn Q, and then describe how the actor parameters are updated. Finally, we discuss how the trainable relaxation parameter is automatically tuned in our setup for exploration and variance minimization. The algorithm is summarized in Algorithm 1.

3.1 Critic Update

We use a Q function parameterized by w, where in our case w are the parameters of a neural network. Unlike DPG, our policy is discrete and allows the Q function to be updated by minimizing the mean squared error (MSE) between Q and a fixed target. To address overestimation bias [19] in the critic update, we employ Double Clipped Q-Learning [7]. The target Q in the MSE now consists of taking the minimum of two Q-functions in the critic update:

$$L(w) = \frac{1}{N}\sum_i (r_i + \gamma \min_{i=1,2} Q'_{\tilde{w}_i}(s_{i+1}, \pi_\theta(s_{i+1})) - Q_w(s_i, a_i))^2 \tag{3}$$

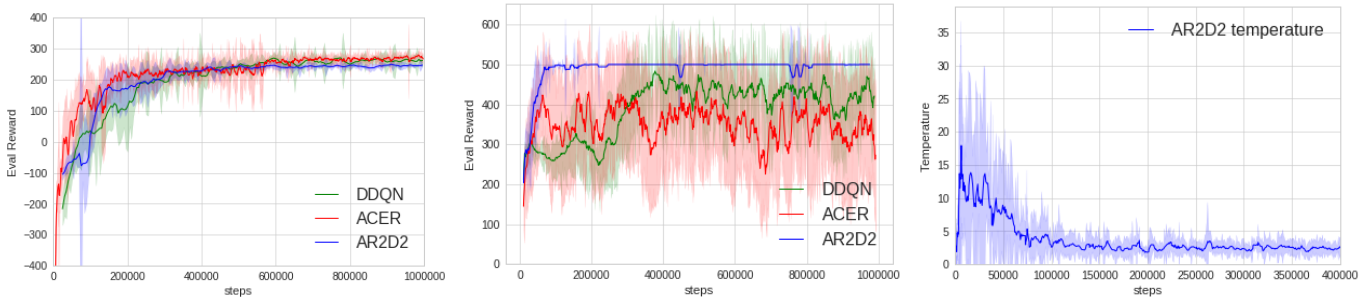
where (s_i, a_i, r_i, s_{i+1}) are a collection of experiences from the environment.

3.2 Actor Update

Expanding the Gumbel-Softmax policy definition from Equation 1 reveals three sets of variables: the categorical probabilities $\{\eta_1, \ldots, \eta_{|A|}\}$, the Gumbel noise $\{g_1, \ldots, g_{|A|}\}$ and the temperature parameter τ. The categorical parameters are implemented with a deep neural network and updated by following the gradient in Equation 2.

3.3 Temperature Update

While the addition of the temperature is added to Gumbel-Softmax as a requirement for being differentiable, its introduction offers a unique opportunity in the reinforcement learning domain to automatically control the balance between



(a) Performance on LunarLander-v2 (b) Performance on CartPole-v1 (c) Temperature auto-tuned in training

Figure 1: **Evaluation performance for three algorithms on (a) LunarLander-v2 and (b) CartPole-v1. (c) shows the behaviour of temperature during learning.** (a) While all three algorithms solve LunarLander (> 200 reward), AR2D2 displays lower variability between random seeds. (b) While ACER and DDQN show high variance for CartPole, AR2D2 converges quickly even when using the same hyperparameters as Lunar Lander. We plot a smoothed mean-return along with standard deviation (shaded) for 5 random seeds. (c) The temperature increases at the start of learning and decays automatically over time.

exploration and exploitation: adjusting the temperature from zero to infinity interpolates the distribution between a deterministic $\arg\max$ and a uniform distribution. The role of τ is then similar to the downstream impact of entropy regularization in that it controls policy stochasticity [20, 21] and avoids the need for external exploration noise often added to off-policy algorithms [16, 2, 19]. Given τ is part of the same computational graph which optimizes η, we formulate two separate gradient updates for τ, one to maxizime discounted return and another to minimize variance.

In order to stabilize the changing policy, we minimize the policy gradient variance w.r.t. τ, as proposed in [9], who optimize the temperature of a relaxed hard threshold control variate. The gradient of the variance in gradient wrto τ is formulated as:

$$\frac{\partial}{\partial \tau}\text{Var}(g(\pi_\eta)) = \frac{\partial}{\partial \tau}\left(\mathbb{E}[g(\pi_\eta)^2] - \mathbb{E}[g(\pi_\eta)]^2\right) = \mathbb{E}\left[2g(\pi_\eta)\frac{\partial g(\pi_\eta)}{\partial \tau}\right] \tag{4}$$

where $g(\pi_\eta)$ is the gradient of Equation 2 w.r.t. the categorical parameters η. The second update takes a gradient ascent step w.r.t. τ in the direction which maximizes Q. Combining the two, we get the following update for τ:

$$\tau_{t+1} = \tau_t + \alpha_\tau^Q[\nabla_a Q(s,a)|_{a=\pi_\theta(s)}\nabla_\tau \pi_\theta(s)] - \alpha_\tau^\sigma \nabla_\tau \text{Var}(g(\pi_\theta)) \tag{5}$$

where α_τ^Q and α_τ^σ are respective learning rates for the gradien updates to maximize Q and minimize the gradient variance from Equation 4. We find that a high α_τ^Q learning rate, allowing the policy to quickly interpolate between exploration and exploitation, is key to quick convergence of the algorithm. The algorithm is summarized in Algorithm 1.

4 Experimental Results

In this section we show the viability of using continuous relaxations by comparing with two state-of-the-art off-policy RL algorithms: Double Deep-Q Learning (DDQN) [19] and ACER [4] on two discrete action environments, LunarLander-v2 and CartPole-v1 [22]. Algorithms are compared based on rollouts of the greedy policy. We tune hyperparameters on LunarLander-v2 and transfer them without modification to CartPole-v1.

All methods solve LunarLander-v2 (Figure 1a). Interestingly, the hyperparameters for AR2D2 found on LunarLander-v2 transferred without modification onto CartPole-v1 unlike ACER and DDQN (Figure 1b) for which we had to fine-tune the learning rate. This suggests that the auto-tuning mechanism might provide increased stability and robustness to hyperparameters in our algorithm. Additionally, we note the remarkable stability of AR2D2 on CartPole (Figure 1b compared to ACER and DDQN, despite the simplicity of the domain. A more robust algorithm would be a strong addition to the current repertoire of RL algorithms and a further exposition of robustness will be left to future work.

In Figure 1c we can see how the temperature parameter increases substantially at the beginning of training, while quickly decreasing around 100,000 steps. An increase in the temperature τ suggests both increased exploration and smoothing of the problem early on during training.

5 Conclusion

In this work, we have shown empirical evidence for using continuous relaxations of discrete random variables in an off-policy policy gradient algorithm. Particularly interesting is the dual purpose of the temperature parameter τ. It controls

both the relaxation and the data collected from the environment, i.e. exploration. Specifically, the relaxation can be seen as a form of smoothing and its relationship to entropy regularization will be explored in future work [21].

In summary, our work has unified discrete and continuous actions in the same off-policy policy gradient algorithm. We expect that other RL algorithms that have previously faced the "differentiabiliy" requirement can successfully take advantage of the relaxation. Future work will consider a more thorough theoretical and empirical investigation of performance as well as the robustness of AR2D2 to hyperparameters.

References

[1] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 2016.

[2] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[3] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[4] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *International Conference on Learning Representations*, 2017.

[5] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[6] Ehsan Imani, Eric Graves, and Martha White. An off-policy policy gradient theorem using emphatic weightings. In *Advances in Neural Information Processing Systems*, 2018.

[7] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

[8] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*, 2017.

[9] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, 2017.

[10] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*, 2017.

[11] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

[12] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *31st International Conference on Machine Learning*, 2014.

[15] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 2017.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[17] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

[18] Emil Julius Gumbel. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series*, 1954.

[19] Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, 2010.

[20] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 1991.

[21] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. *arXiv preprint arXiv:1811.11214*, 2018.

[22] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

6 Appendix

The algorithm derived from the updates in Section 3 is shown below:

Algorithm 1 AR2D2

Input: critic networks Q_{w_1}, Q_{w_2}, and Gumbel-Softmax actor network π_η and π_τ
Input: update actor step d, temperature learning rate α_τ, update weight β
Initialize target networks $w_1' \leftarrow w_1, w_2' \leftarrow w_2, \theta' \leftarrow \theta$
Initialize replay memory $\mathcal{D}$
for episode $= 1$ **to** M **do**
  Initialize s_t
  for $t = 1$ **to** T **do**
    Select action $a = \pi_\theta(s_t)$
    Discretize action a using $\arg\max$ to obtain $\hat{a}$.
    Observe $(r, s_{t+1}) = \text{env}(\hat{a})$
    Append $\mathcal{D}$ with tuple (s_t, a, r, s_{t+1})
    Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{D}$
    $y \leftarrow \begin{cases} r & \text{for terminal state } s \\ r + \gamma \min_{i=1,2} Q_{\tilde{w}_i}(s', a) & \text{for non-terminal state} s \end{cases}$
    Update critics $w_i \leftarrow \arg\min_{w_i} N^{-1} \sum (y - Q_{w_i}(s, a))^2$
    if $t \text{mod} d$ **then**
      Update policy gradients:
      $\nabla_\eta J(\eta) = N^{-1} \sum \nabla_a Q(s, a)|_{a=\pi_\theta(s)} \nabla_\eta \pi_\theta(s)]$
      $\nabla_\tau J(\tau) = N^{-1} \sum \nabla_a Q(s, a)|_{a=\pi_\theta(s)} \nabla_\tau \pi_\theta(s)]$
      $\nabla_\tau \text{Var}(g(\pi_\theta)) = \mathbb{E}[2g(\pi_\theta) \nabla_\tau g(\pi_\theta)]$
      Update target networks:
      $w_i' \leftarrow \beta w_i + (1 - \beta) w_i'$
      $\eta' \leftarrow \beta \eta + (1 - \beta) \eta'$
      $\tau' \leftarrow \beta \tau + (1 - \beta) \tau'$
    end if
  end for
end for

Modelling Individual Differences in Exploratory Strategies: Probing into the human epistemic drive

Nicolas Collignon
School of Informatics
University of Edinburgh
n.collignon@ed.ac.uk

Christopher Lucas
School of Informatics
University of Edinburgh
clucas2@inf.ed.ac.uk

Abstract

People often navigate new environments and must learn about how actions map to outcomes to achieve their goals. In this paper, we are concerned with how people direct their search and trade off between selecting informative actions and actions that will be most immediately rewarding when they are faced with novel tasks. We examine how memory constraints and prior knowledge affect this drive to explore by studying the exploratory strategies of people across four experiments. We find that some people were able to learn new reward structures efficiently, selected globally informative actions, and could transfer knowledge across similar tasks. However, a significant proportion of participants behaved sub-optimally, prioritizing collecting new information instead of maximizing reward. Our evidence suggests this was motivated by two types of epistemic drives: 1) to reduce uncertainty about the structure of the task and 2) to observe new evidence, regardless of how informative they are to the global task structure. The latter was most evident when participants were familiar with the task structure, hinting that the drive to gather knowledge can be independent of learning an abstract representation of the environment. This was not the case when observations did not remain visible to participants, suggesting that participants may adapt their exploratory strategies not only to their environment but also to the computational resources available to them. Our initial modelling results attempt to explain the different cognitive mechanisms underlying human exploratory behaviour across tasks, and are able to capture and explain systematic differences across conditions and individuals.

Keywords: active learning; generalization; exploration-exploitation; heuristics; transfer learning;

1 Introduction

In order to act, plan, and achieve goals, people must learn about their environment and the outcome of possible actions. One reason for human successes in developing new theories and strategies when confronted with new problems is that people are not passive observers. Indeed, children ask informative questions and can adapt their strategies when inquiring about things they don't know [1], and play with new toys in ways that help them disambiguate uncertain causal relationships and gather information [2, 3]. The idea that humans learn and interact with their environment by performing intuitive experiments, maximizing information gain, is a popular one [4, 5, 6, 7].

In this work, we are interested in how people learn to select actions that are most rewarding when faced with a sequence of novel but potentially related tasks. We designed experiments to better understand people's exploration and reward maximizing strategies across a sequence of tasks. Do those strategies evolve over time, as they encounter related tasks? Can people transfer structural knowledge and improve their performance by leveraging similarities between tasks? What is the relationship between people's search strategies, their ability to learn and generalize from observations, and how well they perform?

When faced with new situations, people are often faced with the decision of either gathering more information about the task to improve the quality of their decision, or choosing an action that has been shown to be rewarding [8]. A doctor might, for example, want to run more tests to have a better diagnosis for their patient or give them the treatment they believe will best relieve them from their symptoms. To better understand human decision strategies when dealing with the explore-exploit trade-off, Multi-armed Bandits (MAB) have been used extensively. In these experiments, participants have to select between different possible actions yielding stochastic rewards, so as to maximize rewards. In the real world, an essential part of solving problems lies in discovering the underlying structure of the problem, where each action can be represented as a set of continuous and discrete features. In a Contextual MAB (CMAB), each arm has a set of features that may be informative of the arm's reward distribution. Learning how features relate to rewards allows for an efficient representation of the environment, and enables the learner to generalize to new events.

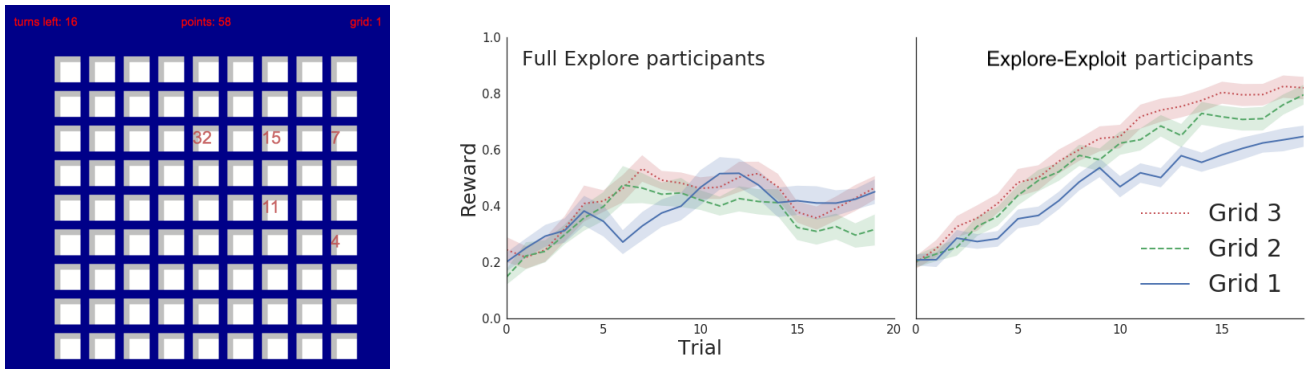
We report on two experiments where people have to find rewarding actions in a sequence of tasks, and where the reward structure is unknown. We compare them to cases where participants were trained on the reward structures prior to the task. We find evidence that some participants selected actions that resolve uncertainty about the underlying structure of the task, and traded off between exploration and exploitation in order to maximize reward. These participants were also able to transfer knowledge across tasks and gradually improved their performance. Conversely, a significant proportion of participants engaged in pure exploratory behavior, consistently preferring to attend novel information rather than maximizing rewards. We highlight the importance of studying individual differences when studying human learners and identify independent factors of epistemic drive that guide human exploration.

2 Experiment 1

We designed our initial experiment to look at how participants adapt to change of reward structure, and detect similarities between tasks, with the hypothesis that people's behavior would be well accounted by Bayesian models. What we find instead is that, contrary to previous studies, the behavior of many participants deviated from those models' predictions.

To better understand this phenomenon, we focused on the first three tasks each participant completed, which shared a similar underlying reward structure. Participants were given a sequence of grids of 9x9 tiles, with each tile corresponding to a possible choice. Participants had to select tiles to maximize their cumulative rewards over 20 choices in each grid. This presents a classical explore-exploit trade-off: Succeeding in the task requires carefully balancing between choosing new tiles to learn about the underlying reward structure or re-selecting tiles that were observed to be rewarding. In each grid, contextual features (x,y) predicted for rewards. When a tile is selected, the reward is displayed for a short period of time and is added to the cumulative score on the current grid. Participants were given no information about the underlying structure of the grid prior to the task, apart from the fact that there may be patterns behind the rewards across tiles. In the game, it is possible to re-select a tile repeatedly, and contrary to traditional bandit tasks, rewards were deterministic for any given tile. This was done to ensure actions were distinctly either exploratory or exploitative (as opposed to a stochastic case, where one could re-select an option to learn about it's volatility.).

Experiment 1 showed that some participants were able to learn the underlying task structure when it was new and traded off between exploration and exploitation to maximize their rewards. These participants transferred knowledge across tasks that shared similarities in their underlying structure. However, we observed that a large proportion of participants had a strong tendency to over-explore, preferring unobserved tiles over known tiles with a high reward value. Twenty-two participants (31 percent) never re-selected tiles more than twice in any of the grids. We call these participants *Full eExplore* (FE) participants. We call the other participants (n=49), that traded off exploration and exploitation, *Explore-Exploit* (EE) participants. We plot the performance of EE and FE participants across all three grids in Figure 1. Further,



(a) Game screenshot

(b) Experiment 1 performance

Figure 1: (a) The grid presented to participants after 5 observations. Note that in Experiment 1, the rewards disappear shortly after a tile has been selected. (b) Performance of FE participants (n=22) and EE participants (n=49) in Experiment 1 across all three grids. The plotted confidence interval corresponds to the standard error (ci=68%).

participants had overall a strong 'local bias' in their sampling. Both EE and FE groups showed this bias, with adjacent tiles selected in 49% of FE participants' exploratory choices and 39% for EE participants.

To explain the large proportion of FE participants, we hypothesized participants may have been driven by wanting to learn more about the reward structure and collect information. This would be consistent with the local search strategies exhibited in other domains such as causal learning [9], category learning [10], or more generally with people's inherent curiosity bias [11, 12]. We hypothesized that this would only be the case for new tasks when participants still had something to learn about the underlying reward structure of the tasks.

3 Experiment 2

Experiment 2 was identical to Experiment 1, but with the reward displayed continuously once a tile has been observed. We added comprehension questionnaires and changed the reward scheme to rule out the alternative explanations about participants' extreme exploratory behavior. We hypothesized that with participants observations remaining visible, the overall reward pattern would be more evident. Thus, participants would be more likely to re-select tiles with high values and perform better than in Experiment 1. Because the underlying structure was more evident, we also assumed fewer participants would engage in *full exploration* behavior, since their curiosity drive would be less pronounced. We also hypothesized that participants would be able to make more globally informative actions (i.e. exploratory selections would be more distant from each other).

Against our expectations, participants were overall more prone to engage in *full exploratory* behavior than in Experiment 1. It could be that participants were further motivated to collect more observations when they remained visible, as the pattern might have been more salient to them and allowed them to learn better. Following our hypothesis that visible observations allowed participants to generalize better, EE participants in Experiment 2 had more global exploratory selections at the beginning of each grid. This could explain their better average performance on the first grid when compared to those in Experiment 1.

4 Experiment 3

In Experiment 3, we tried to understand the large proportion of participants that engaged in full exploratory behavior. After Experiment 1, we hypothesized that this might have been due to an intrinsic epistemic drive in participants. We controlled for several alternative hypotheses, such as memory constraints, unclear instructions, or reward incentives, but this led to more participants engaging in pure exploratory behavior. We designed Experiment 3 to control explicitly for the potential epistemic drive of FE participants. To do this, we explicitly instructed participants about the relationship between a tile's location and the corresponding reward, prior to the task.

By making the structure clear to participants prior to the tasks, our primary prediction for Experiment 3 was that fewer participants would engage in *full exploratory* behavior, since the epistemic reward would be largely attenuated. We also hypothesized there would be weaker or no progress across grids since participants would already be familiar with the reward structure from the first grid. With this training, we predicted participants would be more efficient at finding and re-selecting tiles with high values, and would thus perform better overall than in Experiment 1 and 2. Experiment 3 was set up identically to Experiment 2 except from the addition of a training step where participants were given one practice

grid where all the rewards were continuously displayed, then two further practice grids, similar to the actual task grids, so that they could learn the underlying pattern prior to performing the task.

Contrary to our hypothesis, many participants still engaged in full exploratory behavior. Given this result, we hypothesized that participants might be motivated by observing new rewards rather than learning the underlying reward structure *per se* and that this effect might have been emphasized by the fact that rewards remained visible after having been selected once. Indeed, in Experiment 2, where rewards remained observable, significantly more participants engaged in full-exploratory behavior than in Experiment 1. We designed Experiment 4 to account for both factors of epistemic motivation: 1) wanting to learn about the underlying structure or the location of the maximum, and 2) wanting to observe novel information.

5 Experiment 4

Our main hypothesis for Experiment 4 was that fewer participants would engage in *full exploratory* behavior, since the epistemic reward is attenuated by not having the tiles visible after they have been selected and having training grids prior to the task. We predicted EE participants would perform similarly or slightly worse than in Experiment 3, because of the constraints of not having previous observations visible, but better than in Experiment 1 and 2. We also predicted we would observe little or no transfer effect across grids.

In agreement with our hypothesis, only one participant out of 37 engaged in *Full Exploration*. This was significantly less than in any other condition. This supports the idea that participants' strategies were driven by an epistemic drive which was twofold. First, participants were motivated to reveal the underlying reward structure, e.g., reducing the entropy about the structure of the task, or about the location of the maximum. Participants were less likely to engage in FE behavior in Experiment 4 (known structure and disappearing observations) than Experiment 1 (unknown structure and disappearing observations), and significantly less in Experiment 3 (known structure and visible observations) than Experiment 2 (unknown structure and visible observations). Second, participants were motivated to observe the outcomes of individual actions, with a preference for actions that were local to their last one

Participants' drive to reduce local uncertainty was enhanced by the fact that information became available once it has been observed once. They were engaged less in FE behavior in Experiment 1 (non-visible observations) than Experiment 2 (visible observations), and less in Experiment 4 (non-visible observations) than Experiment 3 (visible observations).

6 Computational Modelling: Initial Results

We are currently investigating how computational models of memory, generalization and search can give us insight into people's representations and strategies when learning in new environments. Besides the important differences across experiments, we are also interested in investigating the differences in behaviour of participants from the same experimental condition. People's explore-exploit strategies have been shown to carry significant differences across individuals [13]. More generally, advances in statistical and modelling tools has led to an increased interest understanding qualitative differences in how people think and act [14].

We outline briefly the different components used in our model to capture different mechanisms of human behavior. To model directed search, we use the predictions of Gaussian Process (GP) with an RBF Kernel. We take a fully Bayesian treatment of the GP kernel hyperparameters, as presented in [15]. GPs have been successful in explaining human function learning phenomena [16, 17], unifying conflicting theories about how humans learn functions. More recently they have also been applied to study decision making in multi-armed bandit problems [18]. We define a greedy weight component that assigns a probability weight to reselect the currently maximum known value. To account for the local bias observed in participants, we use the inverse Manhattan distance (IMD) to the last observation and fit with a softmax temperature parameter to individual participants. We also add a negative weight on previous observations and a random exploratory term (uniform probability for all observations). Models are fit to individual participants by using a Differential Evolution algorithm to maximise the maximum likelihood function. We use an L1 penalty on all weight parameters and an exponential penalty on the local-bias temperature parameter for more interpretable models. We map the resulting models in Figure 2 to highlight clusters of behaviours across all four experiments. Table 1 presents the parameters of cluster centroids obtained after running a Gaussian Mixture Model over all participants, as plotted in Figure 2. The results show that we can obtain interpretable parameters that are consistent with observed the participant behaviors.

7 Conclusion

In this paper, we focused on the behavioural analysis of participants across four experiments to study how people learn to select rewarding actions in a sequence of novel tasks. We found that some participants were able to learn the underlying structure while balancing exploration and exploitation to maximize their rewards across tasks. They improved

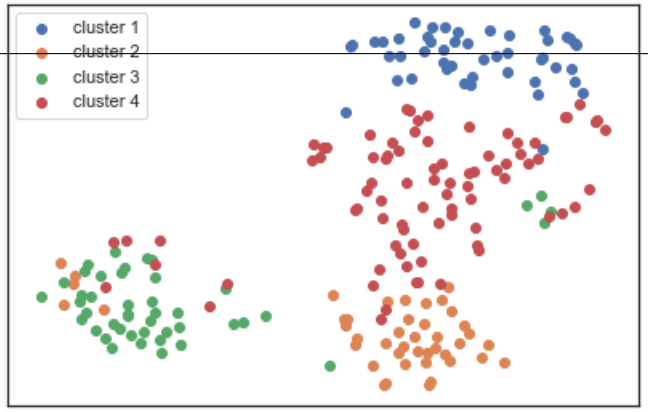


Figure 2: t-SNE visualisation of MLE parameters for individual participants across all 4 experiments. Clusters are obtained via a Gaussian Mixture Model. Cluster centroids are reported in Table 1.

	α directed search ($E[x]$ under the GP)	β global search (σ^2 under GP)	Greedy weight (reselecting max-known)	local-bias weight	local-bias temperature	dampen previous observations	random exploration
cluster 1	0.3	0.06	0.22	0.03	75.13	0.15	0.24
cluster 2	0.02	0	0.12	0.35	26.15	0.3	0.21
cluster 3	0.1	0	0.08	0.51	7.21	0.03	0.28
cluster 4	0.22	0.04	0.19	0.19	1.57	0.14	0.18

Table 1: Parameters of cluster centroids of Gaussian Mixture Model. Weight parameters are normalised (i.e. all but the local-bias temperature). These results show that we can obtain interpretable parameters that are consistent with observed the participant behaviors. E.g. Cluster 1 corresponds to EE participants with global exploration, cluster 3 corresponds to FE participants with a strong local bias.

their performance from one task to the next by transferring abstract knowledge about their environment. However, consistently across tasks, we observed that a significant proportion of participants engaged in purely exploratory behavior, largely ignoring the reward incentive. We showed that this behavior could be manipulated by controlling the availability of information as the learner selected actions, and by giving prior knowledge before participants engaged with the task. We suggest that people are motivated by two types of epistemic drives: 1) to reduce uncertainty and learn about the structure of the task and 2) to observe new evidence, regardless of its informativeness about the global task structure. In our study, we highlight that studying individual differences amongst participants can help us better understand the complex mechanisms at play during active learning in new environments.

References

[1] Azzurra Ruggeri and Tania Lombrozo. Learning by asking: how children ask questions to achieve efficient search. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, pages 1335–1340, 2014.

[2] Laura Schulz and Elizabeth Baraff Bonawitz. Serious fun: preschoolers engage in more exploratory play when evidence is confounded. *Developmental psychology*, 43(4):1045, 2007.

[3] Claire Cook, Noah D Goodman, and Laura E Schulz. Where science starts: Spontaneous experiments in preschoolers? exploratory play. *Cognition*, 120(3):341–349, 2011.

[4] Anna Coenen, Jonathan D Nelson, and Todd Gureckis. Asking the right questions about human inquiry. 2017.

[5] Todd M Gureckis and Douglas B Markant. Self-directed learning: A cognitive and computational perspective. *Perspectives on Psychological Science*, 7(5):464–481, 2012.

[6] Jonathan D Nelson. Finding useful questions: On bayesian diagnosticity, probability, impact, and information gain. *Psychological review*, 112(4), 2005.

[7] Alison Gopnik, Clark Glymour, David M Sobel, Laura E Schulz, Tamar Kushnir, and David Danks. A theory of causal learning in children: causal maps and bayes nets. *Psychological review*, 111(1):3, 2004.

[8] Thomas T Hills, Peter M Todd, David Lazer, A David Redish, Iain D Couzin, Cognitive Search Research Group, et al. Exploration versus exploitation in space, mind, and society. *Trends in cognitive sciences*, 19(1):46–54, 2015.

[9] Neil R Bramley, David A Lagnado, and Maarten Speekenbrink. Conservative forgetful scholars: How people learn causal structure through sequences of interventions. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 41(3):708, 2015.

[10] Douglas B Markant, Burr Settles, and Todd M Gureckis. Self-directed learning favors local, rather than global, uncertainty. *Cognitive science*, 40(1):100–120, 2016.

[11] Celeste Kidd and Benjamin Y Hayden. The psychology and neuroscience of curiosity. *Neuron*, 88(3):449–460, 2015.

[12] Alison Gopnik. Explanation as orgasm. *Minds and machines*, 8(1):101–118, 1998.

[13] Mark Steyvers, Michael D Lee, and Eric-Jan Wagenmakers. A bayesian analysis of human decision-making on bandit problems. *Journal of Mathematical Psychology*, 53(3):168–179, 2009.

[14] Daniel J Navarro, Thomas L Griffiths, Mark Steyvers, and Michael D Lee. Modeling individual differences using dirichlet processes. *Journal of mathematical Psychology*, 50(2):101–122, 2006.

[15] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[16] Christopher G Lucas, Thomas L Griffiths, Joseph J Williams, and Michael L Kalish. A rational model of function learning. *Psychonomic bulletin & review*, 22(5):1193–1215, 2015.

[17] Eric Schulz, Josh Tenenbaum, David K Duvenaud, Maarten Speekenbrink, and Samuel J Gershman. Probing the compositionality of intuitive functions. In *Advances In Neural Information Processing Systems*, pages 3729–3737, 2016.

[18] Charley M Wu, Eric Schulz, Maarten Speekenbrink, Jonathan D Nelson, and Björn Meder. Generalization guides human exploration in vast decision spaces. *Nature Human Behaviour*, 2(12):915, 2018.

Remediating Cognitive Decline with Cognitive Tutors

Priyam Das
University of California, Irvine
Irvine, CA
priyam.das@uci.edu

Fred Callaway
Princeton University
Princeton, NJ
fredcallaway@princeton.edu

Thomas L. Griffiths
Princeton University
Princeton, NJ
tomg@princeton.edu

Falk Lieder
Max Planck Institute for Intelligent Systems
Tübingen, Germany
falk.lieder@tuebingen.mpg.de

Abstract

As people age, their cognitive abilities tend to deteriorate, including their ability to make complex plans. To remediate this cognitive decline, many commercial brain training programs target basic cognitive capacities, such as working memory. We have recently developed an alternative approach: intelligent tutors that teach people cognitive strategies for making the best possible use of their limited cognitive resources. Here, we apply this approach to improve older adults' planning skills. In a process-tracing experiment we found that the decline in planning performance may be partly because older adults use less effective planning strategies. We also found that, with practice, both older and younger adults learned more effective planning strategies from experience. But despite these gains there was still room for improvement – especially for older people. In a second experiment, we let older and younger adults train their planning skills with an intelligent cognitive tutor that teaches optimal planning strategies via metacognitive feedback. We found that practicing planning with this intelligent tutor allowed older adults to catch up to their younger counterparts. These findings suggest that intelligent tutors that teach clever cognitive strategies can help aging decision-makers stay sharp.

Keywords: aging; planning; cognitive training; cognitive plasticity

Acknowledgements

This work was supported by grant number ONR MURI N00014-13-1-0341 and a grant from the Templeton World Charity Foundation to TLG.

1 Introduction

Many cognitive abilities deteriorate with normal aging, including planning. Commercial brain training programs promised to remediate this cognitive decline by training basic cognitive capacities – especially working memory. But they have often failed to live up to their promises (*A consensus on the brain training industry from the scientific community*, 2014). More effective methods for combating this decline or even improving planning abilities have yet to be discovered. One new approach could be to discover and teach people cognitive strategies that make the best possible use of their bounded cognitive resources (Lieder, Callaway, Das, et al., 2019; Lieder et al., 2018; Lieder, Krueger, Callaway, & Griffiths, 2017).

Previous studies have found that older adults have trouble formulating plans and updating them in the light of feedback (Allain et al., 2005; Sorel & Pennequin, 2008). We hypothesized that the reason why older adults perform worse is that their planning strategies are less effective than those of younger adults. If this is the case, then it should be possible to mitigate this aspect of cognitive decline by teaching older adults better planning strategies. Here we investigate this hypothesis using the intelligent cognitive tutor we developed in previous work (Lieder, Callaway, Das, et al., 2019; Lieder et al., 2018, 2017). In Experiment 1, we characterized the planning strategies used by people of different age groups in order to determine whether age affects the types of planning strategies used. In Experiment 2, we investigated whether cognitive tutoring can help close the performance-gap between younger and older adults. Our results suggest that cognitive tutoring is a promising approach that should be explored as an intervention for improving people's decision-making competency and remediating cognitive decline.

2 Experiment 1

2.1 Methods

We recruited participants younger than 25 years old to form our younger adults group ($19-24$ y.o., median $=23$, $n=49$) and adults older than 47 years old to form our older adults group ($48-70$ y.o., median $=52$, $n=29$). The experiment was conducted online via Amazon Mechanical Turk. In the experiment, participants completed 30 trials of the Mouselab-MDP paradigm (Callaway, Lieder, Krueger, & Griffiths, 2017) with a three-step route planning task. On each trial, participants were shown a map of gray circles (Figure 1) and instructed to move the spider in the middle to one of the outermost nodes, picking up the rewards hidden along the way. For each trial, rewards are independently drawn from discrete uniform distributions; in the first step the possible values were $\{-4,-2,+2,+4\}$; in the second step the possible values were $\{-8,-4,+4,+8\}$; and in the third step the possible values were $\{-48,-24,+24,+48\}$. Participants could uncover rewards beforehand by clicking on the gray circles and paying a cost of -1 for each reveal. Participants were instructed to maximize their rewards and were incentivized with a monetary bonus based on their in-game score.

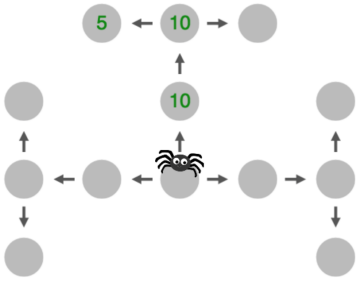


Figure 1: A typical Mouselab-MDP trial used in Experiment 1 and the control condition of Experiment 2. Some of the rewards have already been revealed by the participant.

We use the clicks our participants made to infer which kind of planning strategy they used. We considered six different planning strategies: depth-first search, breadth-first search, best-first search, progressive deepening, the optimal planning strategy, and an impulsive strategy that chooses randomly. Depth-first search explores a single path at a time – from its beginning to its end. Once it reaches the end of this path, it returns to the most recent unexplored fork in that path and continues exploring until all nodes have been inspected. Breadth-first search explores the first nodes of all possible paths, then the second nodes, and so on until all paths have been explored. Best-first search explores paths in the order of highest expected sum of rewards. Progressive deepening is a strategy proposed by Newell and Simon (1972) and is similar to depth-first search. The main difference is that after exploring a path in its entirety, progressive deepening skips back to the starting node, treating branches as part of another path for later exploration. Callaway et al. (2018) found that the optimal strategy for the task environment used in this experiment is to first set a goal by evaluating potential final destinations. As soon as inspecting a potential final destination uncovers the highest possible reward ($+48$), the optimal strategy selects the path that leads to it and terminates planning. If multiple potential final destinations are good (i.e., $+\$24$) then the optimal strategy collects additional information about the paths leading to those promising potential final destinations starting with the nodes right before a potential final destination.

2.2 Modeling Strategies

We modeled participants' click sequences as a combination of following one of the six strategies described above and some random moves. Formally, the probability of making click c when following strategy k is defined as

$$(1-\epsilon)\cdot\sigma(c;V_{b,M_k},\tau)+\epsilon\cdot\text{Uniform}(c;C_b) \quad (1)$$

where the first term, $\sigma(c;V_{b,M_k},\tau)$, is a softmax over the possible clicks c in state b when following strategy k and τ is the temperature parameter. The second term, $\text{Uniform}(c;C_b)$, can account for actions that are inconsistent with strategy k; the probability of such "random clicks" is modeled by a uniform distribution over all possible clicks and the action of stopping planning. Finally, ϵ is the probability that a random click will be made.

The random strategy can therefore be modeled by the second term alone. The systematic behavior of the other strategies was modeled in terms of the values $V_{b,M}(c)$ they assign to different clicks c and the decision to terminate planning. For example, in the depth-first search model, the preference function $V_{b,DFS}(c)$ is the depth of the node inspected by click c if that node lies on a partially explored path and a large negative value otherwise. As a result, deeper nodes are prioritized and partially explored paths will be explored to the end before others are considered. In the optimal strategy model, the value assigned to $V_{b,O}$ is given by the optimal solution to the problem of deciding how to plan. In previous work, we formalized this problem as a meta-level Markov Decision Process and computed its solution for the environment used in this study using backwards induction (Callaway et al., 2018). Aside from the random and optimal strategy models, all of our strategy models also capture previous findings that people often act as soon as they have identified an alternative they deem good enough (i.e., satisficing; Simon, 1956) and tend to stop considering a course of action when they realize it would entail a large loss at one point or another (i.e., pruning; Huys et al., 2012). To model satisficing and pruning, our models include two free parameters for the participant's aspiration level and pruning threshold respectively. When the expected reward for terminating in belief state b exceeds the aspiration level, then our models assign a very large value to the terminate planning action. Conversely, if the expected sum of rewards for any path falls below the pruning threshold, then clicks on the remaining unobserved nodes on that path are assigned a large negative value such that the strategy was discouraged from continuing to explore that unprofitable path. We fit all models to each trial for each participant using maximum likelihood estimation for all model parameters (i.e., τ, ϵ, and the thresholds for satisficing and pruning). We then performed model comparisons using the Bayesian Information Criterion (Schwarz et al., 1978) to determine which strategy each participant is most likely to have used on each trial.

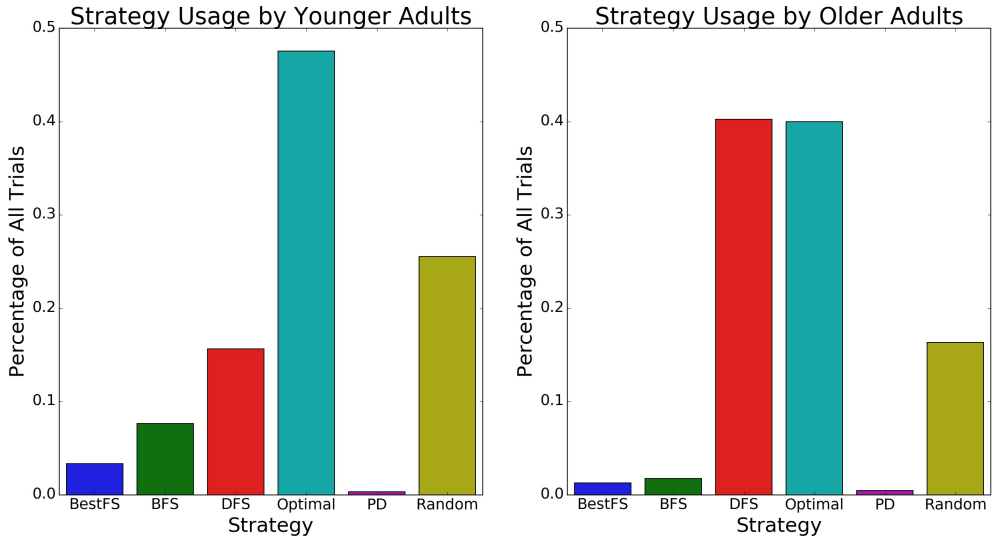


Figure 2: Strategy usage frequencies for younger adults versus older adults over all trials. The strategies we modeled are (from left to right): Best-first search, breadth-first search, depth-first search, optimal, progressive deepening, and random.

2.3 Results

In Experiment 1, we found that older adults differed significantly from younger adults in how often they used each of the six planning strategies introduced above ($\chi^2(5) = 205.43, p < .001$). While both age groups used the optimal strategy the most, older adults also favored the depth-first search strategy, using it almost as much as the optimal strategy (Figure 2).

Taking a look at how participants' strategy usage evolved over time indicates that older adults were adopting the optimal strategy later in the experiment compared to younger adults (Figure 3). However, by the end of the experiment, the older adults were still not using the optimal strategy as frequently as the younger adults (avg. frequency in the last five trials: 69.0% vs. 58.6%, $\chi^2(1) = 3.86, p < 0.05$). We also found that older adults were performing worse on the task compared to younger adults, even after discovering the optimal strategy on their own (avg. score in the last five trials: 24.7 vs. 37.4, $t(76) = -2.87, p < 0.01$). This is consistent with our expectation that using the optimal strategy less than another group will lead to lower scores. If this difference in strategy usage is due to older participants being unaware of the existence of the optimal strategy, then it should be possible to remedy their deficits by teaching them the optimal strategy using our cognitive tutor. We tested this hypothesis in Experiment 2.

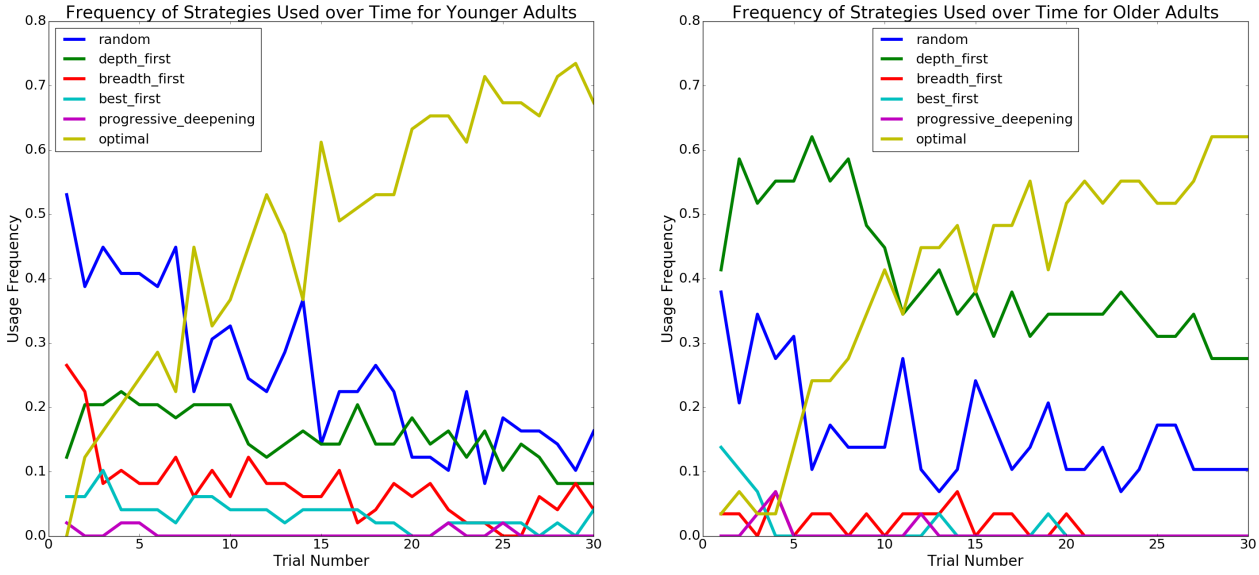


Figure 3: The frequencies of strategy usage for every trial in the experiment for younger adults (left) and older adults (right). The strategies shown are best-first search (cyan), breadth-first search (red), depth-first search (green), optimal (yellow), progressive deepening (magenta), and random (dark blue).

3 Experiment 2

3.1 Methods

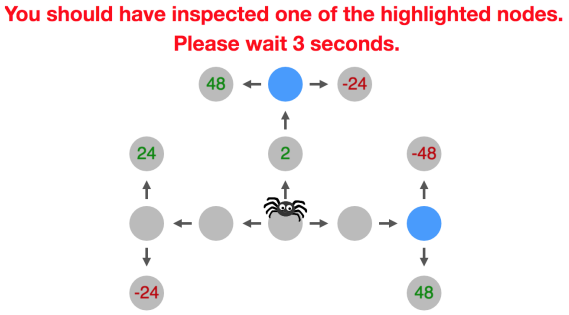


Figure 4: Example feedback from the cognitive tutor in the training phase of Experiment 2.

For Experiment 2, we recruited and sorted participants into two groups: younger than 25 (median $= 22$ years, $n = 41$) and older than 47 (median $= 53$ years, $n = 37$). We conducted the experiment via Amazon Mechanical Turk. Participants were randomly assigned to either train with the cognitive tutor (feedback condition: $18-69$ y.o, $n_{young} = 24, n_{old} = 23$) or to practice the task on their own (control condition: $18-68$ y.o, $n_{young} = 17, n_{old} = 14$). Participants in the control condition performed 30 trials of the Mouselab-MDP task described in the Methods section of Experiment 1. Participants in the feedback condition were first given 15 trials where they practiced the Mouselab-MDP task while receiving our cognitive tutor's optimal metacognitive feedback (Lieder, Callaway, Das, et al., 2019; Lieder et al., 2018, 2017). As illustrated in Figure 4, the tutor's feedback comprised i) a delay penalty whose duration was proportional to how suboptimal the participant's planning operation was, and ii) a visual demonstration of what the optimal planning strategy described above would have done differently. The feedback thereby supported both reinforcement learning and learning from

demonstrations. Participants were then given 15 test trials of Mouselab-MDP without any feedback, identical to the trials given to the control group.

3.2 Results

Consistent with Experiment 1, we found that older people ($Mean = 23.23, SEM = 2.16$) performed worse in the first half of the experiment than younger people ($Mean = 29.40, SEM = 2.33$; $F(1, 1162) = 18.10, p < 0.001$).

Encouragingly, practicing with our cognitive tutor was effective at improving decision-making skills regardless of age. Specifically, older and younger adults who practiced with the cognitive tutor scored significantly higher in the test block than their counterparts in the control condition ($t(39) = 3.31, p < 0.01$ and $t(35) = 3.56, p < 0.01$ respectively). Even more encouragingly, for older people the benefit of training with our cognitive tutor was so large that they did not only catch up to younger people but even scored significantly higher than younger people who had practiced the task on their own ($t(38) = 2.41, p < 0.05$). Furthermore, it appears that older adults benefited more from the cognitive tutor than younger adults. According to a three-way ANOVA the advantage of young people gradually vanished over time in both conditions ($\beta_{\text{trial}\times\text{young}} = -0.13, F(1, 2332) = 5.21, p < 0.05$) and was more pronounced in the training block than in the test block ($\beta_{\text{training block}\times\text{young}} = 5.55, F(1, 1406) = 7.36, p < 0.01$) in the feedback condition. This indicates that older adults were catching up to younger adults over time. As a consequence, we could no longer detect a significant difference between younger versus older adults ($t(45) = -1.02, p = 0.31$) after 15 trials of training with the cognitive tutor. The results suggest that cognitive tutors can help older adults catch up to the younger generations – and sometimes even overtake them.

4 Discussion

Why do older people have a harder time making complex plans and how can we help aging adults retain their planning skills? In Experiment 1, we found that older adults' decision making skills appear to be limited by their reliance on sub-optimal planning strategies. In Experiment 2, we found that this deficit can be remedied by letting older adults practice planning with a cognitive tutor that teaches them an optimal planning strategy via metacognitive feedback. While both older and younger adults benefited from cognitive tutoring, older adults benefited more. As a result, cognitive training with our intelligent tutor decreased the performance gap between older and younger adults.

Our findings suggest that cognitive tutoring could be a promising new approach to remediating cognitive decline. We recently found that using our cognitive tutors improves performance on a near-transfer task (Lieder, Callaway, Jain, et al., 2019), and we plan to investigate whether this benefit transfers to daily life. Developing cognitive tutors for a wide range of different cognitive skills and tailoring them to the needs of various age groups and (psychiatric) populations are exciting directions for future research.

References

Allain, P., Nicoleau, S., Pinon, K., Etcharry-Bouyx, F., Barré, J., Berrut, G., … Le Gall, D. (2005). Executive functioning in normal aging: A study of action planning using the zoo map test. *Brain and cognition*, *57*(1), 4–7.

Callaway, F., Lieder, F., Das, P., Gul, S., Krueger, P. M., & Griffiths, T. L. (2018). A resource-rational analysis of human planning. In *Proceedings of the 40th annual conference of the cognitive science society.*

Callaway, F., Lieder, F., Krueger, P. M., & Griffiths, T. L. (2017). Mouselab-MDP: A new paradigm for tracing how people plan. In *The 3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making, Ann Arbor, MI, USA.* Retrieved from `https://osf.io/vmkrq/`

A consensus on the brain training industry from the scientific community (Online Statement). (2014). Max Planck Institute for Human Development and Stanford Center on Longevity. (Retrieved from http://longevity3.stanford.edu/blog/2014/10/15/the-consensus-on-the-brain-training-industry-from-the-scientific-community/ on March 1 2019)

Huys, Q. J., Eshel, N., O'Nions, E., Sheridan, L., Dayan, P., & Roiser, J. P. (2012). Bonsai trees in your head: how the pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS computational biology*, *8*(3), e1002410.

Lieder, F., Callaway, F., Das, P., Gul, S., Krueger, P., & Griffiths, T. L. (2019). *An intelligent feedback mechanisms for teaching people optimal planning strategies.* (Manuscript in preparation)

Lieder, F., Callaway, F., Jain, Y. R., Krueger, P. M., Das, P., Gul, S., & Griffiths, T. L. (2019). A cognitive tutor for helping people overcome present bias. In *The 4th Multidisciplinary Conference on Reinforcement Learning and Decision-Making, Montréal, QC, CA.*

Lieder, F., Callaway, F., Krueger, P. M., Das, P., Griffiths, T. L., & Gul, S. (2018). Discovering and teaching optimal planning strategies. In *The 14th biannual conference of the German Society for Cognitive Science, GK.*

Lieder, F., Krueger, P. M., Callaway, F., & Griffiths, T. L. (2017). A reward shaping method for promoting metacognitive learning. In *The 3rd Multidisciplinary Conference on Reinforcement Learning and Decision-Making, Ann Arbor, MI, USA.*

Newell, A., Simon, H. A., et al. (1972). *Human problem solving* (Vol. 104) (No. 9). Prentice-Hall Englewood Cliffs, NJ.

Schwarz, G., et al. (1978). Estimating the dimension of a model. *The annals of statistics*, *6*(2), 461–464.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological Review*, *63*(2), 129–138. doi: 10.1037/h0042769

Sorel, O., & Pennequin, V. (2008). Aging of the planning process: The role of executive functioning. *Brain and cognition*, *66*(2), 196–201.

Contrasting the effects of prospective attention and retrospective decay in representation learning

Guy Davidson
College of Computational Sciences
Minerva Schools at KGI
San Francisco, CA 94103
guy@minerva.kgi.edu

Angela Radulescu
Department of Psychology
Princeton University
Princeton, NJ 08544
angelar@princeton.edu

Yael Niv
Department of Psychology &
Princeton Neuroscience Institute
Princeton University
Princeton, NJ 08544
yael@princeton.edu

Abstract

Previous work has shown that cognitive models incorporating passive decay of the values of unchosen features explained choice data from a human representation learning task better than competing models [1]. More recently, models that assume attention-weighted reinforcement learning were shown to predict the data equally well on average [2]. We investigate whether the two models, which suggest different mechanisms for implementing representation learning, explain the same aspect of the data, or different, complementary aspects. We show that combining the two models improves the overall average fit, suggesting that these two mechanisms explain separate components of variance in participant choices. Employing a trial-by-trial analysis of differences in choice likelihood, we show that each model helps explain different trials depending on the progress a participant has made in learning the task. We find that attention-weighted learning predicts choice substantially better in trials immediately following the point at which the participant has successfully learned the task, while passive decay better accounts for choices in trials further into the future relative to the point of learning. We discuss this finding in the context of a transition at the "point of learning" between explore and exploit modes, which the decay model fails to identify, while the attention-weighted model successfully captures despite not explicitly modeling it.

Keywords: reinforcement learning, representation learning, decay, selective attention, behavioral modeling, Dimensions Task, model comparison, trial-by-trial analysis

Acknowledgements

This project was funded by grant W911NF-14-1-0101 from the Army Research Office to YN. The authors wish to thank the Princeton Neuroscience Institute Summer Internship Program that facilitated this collaboration.

1 Introduction

Previous work on representation learning in humans has investigated a role for selective attention in dynamically shaping task representations [1], [2]. Computational models of this process of carving a task into its constituent states have suggested two different mechanisms for implementing selective attention to different features of environmental stimuli: a feature-level reinforcement learning model with decay of values of features of unchosen options (FRLdecay), and an attention-weighted feature-level reinforcement learning model (awFRL). These two models offer conceptually different accounts of selective attention: decay is retrospective, incrementally forgetting previously learned values if options are not chosen again. In contrast, attentional filtering can be seen as prospective, modulating what is learned now for future use, in line with [3], [4].

Even though these models posit different mechanisms, previous work has shown that, on average, they perform equally well at predicting participants' trial-by-trial choices on a multidimensional bandit task called the "Dimensions Task" (see below). Here, we asked which model better accounts for choices on the Dimensions Task as a function of the participant's stage of learning. If the two models map onto the same cognitive process, we would expect them to be indistinguishable in terms of the likelihood they assign to choices at every stage of the learning process. Conversely, if they capture different aspects of the cognitive process, they should account well for different choices. First, we implement a new model, awFRLdecay, which includes both a decay mechanism and attentional weights, and find it predicts participants' choices better than either FRLdecay or awFRL. This suggests that the two mechanisms capture different components of representation learning. We further investigate this finding using a novel trial-by-trial model comparison analysis which takes into account the participant's "point of learning" when comparing likelihoods. We find that the awFRL model best explains behavior around the time when participants learn the correct task representation, while the FRLdecay model best captures choices further beyond the "point of learning." These findings suggest distinct roles for passive decay of feature weights and selective attention in shaping task representations, and demonstrate the utility of moving beyond average likelihoods when performing model comparison.

2 Task

We reanalyzed data from [2]. Twenty-five human participants were tasked with learning which of nine features was more predictive of reward. Participants played 24 "games" consisting of 25 trials each. On each trial of a game, participants chose between three columns, each comprised of a face, a landmark, and a tool. All features were visible on every trial, but feature combinations within a column varied from trial to trial. The target feature randomly changed between games. Participants had full prior knowledge of the task's generative model. That is, they received instructions in the beginning regarding the reward contingencies, and changes in the target feature (i.e., end of games) were signalled explicitly ("New game starting").

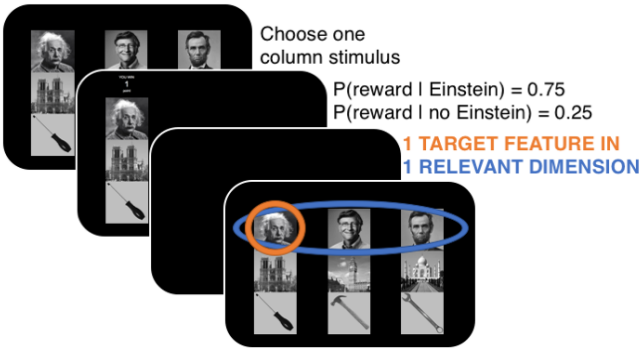


Figure 1: Dimensions Task. On each trial, the participant is presented with three options (columns of features), each including a face, a landmark, and a tool ("dimensions"). After choosing one option, the participant receives feedback, and proceeds to the next trial. One dimension is relevant for determining reward, and within it, one feature rewards with $p = 0.75$ while the other two features reward with $p = 0.25$. The participant does not know *a priori* which dimension is relevant for reward, and which feature is the target feature, and must learn these from trial and error

3 Models Compared

We compared models using leave-one-game-out cross-validation, maximizing the total log-likelihood of the participants' choices. As each participant performed a total of twenty-four games of the task, we held one game out at a time, fit the model on the remaining twenty-three games, and evaluated performance on the held-out game.

3.1 Feature RL with Decay (FRLdecay)

The FRLdecay model introduced in [1] assumes the participant learns a feature weight for each of the nine unique features. We denote these feature weights w and sum to compute a value for each column. Choice likelihood is modeled as a noisy softmax. During learning, the weights of features in the chosen column are updated according to a temporal-difference learning rule. Additionally, the values for features in unchosen columns decay toward zero. The addition of decay implements a retrospective form of selective attention: on any given trial, the model learns about all chosen features equally, later unlearning (decaying) the values of those features that are not chosen again. That is, instead of predicting what should be learned on each trial, the model learns and then forgets values that were learned erroneously.

We denote the ith column on the current trial as S_i, with the feature in each dimension (row) $d \in \{1, 2, 3\}$ accessed as $S_i[d]$, which indexes into the weights vector, $w(S_i[d])$. Column values (analogous to action values, as choices were of

columns) were computed as the sum over the weights of features in the three dimensions. The probability of choosing column c is modeled using a noisy softmax over the values with an inverse temperature parameter β:

$$V(S_i) = \sum_d w(S_i[d]) \tag{1}$$

$$\pi(c) = \frac{e^{\beta V(S_c)}}{\sum_i e^{\beta V(S_i)}} \tag{2}$$

We assume a standard reinforcement learning update rule operating at the feature level. At each time point, we compute the reward prediction error δ from the reward R and the value assigned to the chosen column, $\delta = R - V(S_c)$, and update each chosen feature using a learning rate η. For the unchosen columns, we decay all feature values toward zero with decay rate λ:

$$\text{For all } d: \ w(S_c[d]) = w(S_c[d]) + \eta\delta \tag{3}$$

$$\text{For all } d, i \neq c: \ w(S_i[d]) = (1-\lambda)w(S_i[d]) \tag{4}$$

3.2 Attention weighted feature RL (awFRL)

The attention-weighted feature RL model described in [2] includes empirically-derived dimensional attention weights as a direct measure of selective attention. These weights were computed in two ways: from eye position data, by binning looking time to each dimension within a trial; and from fMRI decoding of information in face-, landmark-, and tool-selective areas of the human cortex. The two measures of attention were then combined to form a single, empirically measured attentional weight for each dimension (d) on each trial, which we denote $\phi[d]$ (see [2] for full details).

The attention weights modified both the value computation and update rule of the FRL model. First, they biased value computation towards features in the attended dimensions. Next, the attention weights also biased the feature weight update, modeling increased attention to a particular dimension as a higher learning rate for features in that that dimension:

$$V(S_i) = \sum_d \phi[d] w(S_i[d]) \tag{5}$$

$$\text{For all } d: \ w(S_c[d]) = w(S_c[d]) + \eta\phi[d]\delta \tag{6}$$

As in the FRLdecay model, choice probabilities followed a noisy softmax distribution on column values.

3.3 Combined model: attention weighted feature RL with decay (awFRLdecay)

This model combines the above models, including both the attention-weighted value computation and chosen feature value updates (from [2]), as well as the decay of unchosen features towards zero (from [1]). We can formally describe the computations on each trial as:

$$V(S_i) = \sum_d \phi[d] w(S_i[d]) \quad \text{Attention-weighted value computation} \tag{7}$$

$$\pi(c) = \frac{e^{\beta V(S_c)}}{\sum_i e^{\beta V(S_i)}} \quad \text{Choice likelihood} \tag{8}$$

$$\text{For all } d: \ w(S_c[d]) = w(S_c[d]) + \eta\phi[d]\delta \quad \text{Attention-weighted feature weight updates} \tag{9}$$

$$\text{For all } d, i \neq c: \ w(S_i[d]) = (1-\lambda)w(S_i[d]) \quad \text{Unchosen feature decay (not attention-weighted)} \tag{10}$$

4 Results

As previously noted, the FRLdecay model and the awFRL model explained participants' choices equally well on average (Figure 2A). The new, combined, awFRLdecay model improved goodness of fit over both previous models, suggesting that retrospective decay and prospective biasing by attention explain different components of the variability in participants' choices. To test this hypothesis more directly, we next compared these models on a trial-by-trial basis, rather than averaging over all trials within each game (Figure 2B). We found that awFRL explained choices in the first half of each game better, while FRLdecay provided a better fit to choices in the later part of each game.

Motivated by the insight that the meaningful indexing within a game is not the absolute trial index, but the one relative to the participant's successful learning of the task, we repeated the trial-by-trial analysis, aligning data around the last trial in which the participant did not choose the target feature, i.e. the "point of learning"(Figure 2C). We found that the awFRL model predicted choices substantially better immediately following the point of learning, while the FRLdecay

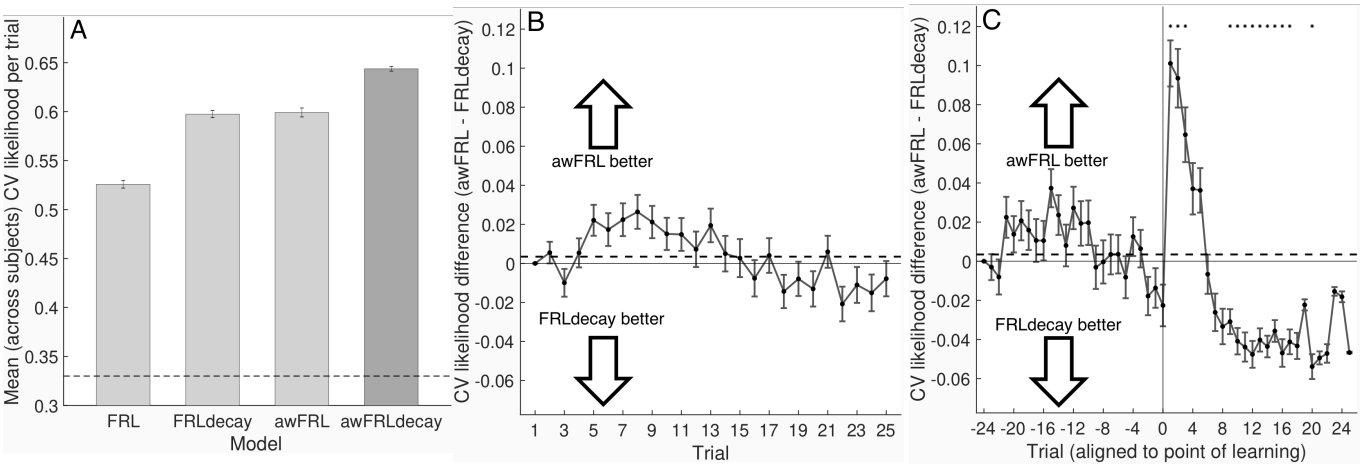


Figure 2: Trial-by-trial model comparison. A: Model comparison showing the mean cross-validated (CV) likelihood of participant's choices per trial. The FRLdecay (best fitting in [1]) and awFRL (best fitting in [2]) models both explain the data equally well, and better than a baseline FRL model (equations (1)-(3)). The combined model surpassed both previous models (paired-sample t tests, $t(24) = 8.998$, $p < .001$; $t(24) = 12.374$, $p < .001$, for the FRLdecay and awFRL models, respectively). **B**: Each bin shows the the average likelihood difference as a function of trial, subtracting the FRLdecay predicted likelihood from the awFRL predicted likelihood. **C**: The same trial-by-trial likelihood comparison, aligned to the last trial in each learned game in which the participant did not choose the target feature (the "point of learning"). Error bars: standard error of the mean (SEM). *: $p < 0.001$.

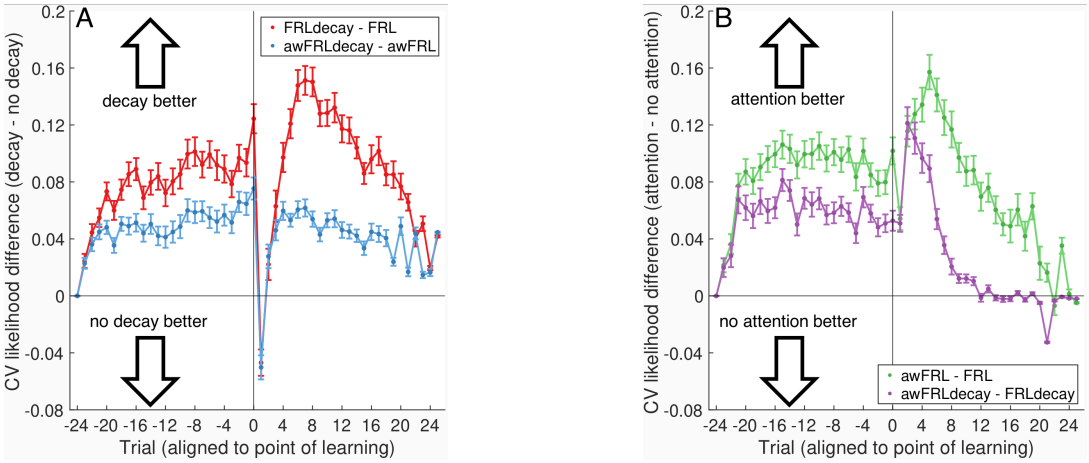


Figure 3: Isolating the effects of introducing decay and attention. A: The difference in choice likelihoods between FRLdecay and FRL (red) and awFRLdecay and awFRL (blue) aligned to the point of learning suggests that models that include retrospective decay lag in capturing the participant's point of learning. **B**: The difference in choice likelihoods between awFRL and FRL (green) and awFRLdecay and FRLdecay (purple) suggests that models that include measured prospective attention weights perform best shortly following the point of learning.

model performed better on the remainder of the game. Interestingly, both models fit the data to an equal extent before the point of learning, that is, during the representation learning phase (see discussion).

To dissociate whether differences in likelihood around the point of learning are due to awFRL being a better predictor of participants' choices, or FRLdecay being a worse predictor, Figure 3 shows the same trial-by-trial analysis as before, separating the effect of adding decay (A) and attention (B) to the models. Adding decay improved goodness of fit throughout, except for the trial immediately following the "point of learning," where decay models predicted choice substantially worse than models omitting decay. Conversely, models including attention weights predicted choices considerably better for a few trials after the "point of learning". Both model components improved goodness of fit before the point of learning.

5 Discussion

We compared the performance of two competing reinforcement learning models on a human representation learning task. The models, introduced in previous work, posit conceptually different mechanisms, one suggesting retrospective

"forgetting" of values learned in error, and the other employing prospective attention-gated learning of only those values that are expected to be useful. Nevertheless, the two models predict choice data equally well on average. The improved goodness of fit when combining both processes suggests that these models may account for different components of the variance in the choice data. We uncovered differences in predictive likelihood between the models using insight into the structure of the task: once participants successfully learn the correct task representation, they terminate a game with a streak of correct choices, transitioning from exploring potential representations to exploiting the correct one. By aligning data to the "point of learning," we recovered diverging predictive capacities of the FRLdecay and awFRL models. The retrospective FRLdecay model lags in capturing behavior at the explore-exploit transition, while the prospective awFRL attention model excels in predicting choices immediately following the transition. Together, these findings provide a more complete account of the complementary role of these mechanisms in enabling representation learning in multidimensional reinforcement learning tasks.

This transition from exploration to exploitation, and the models' success (or lack thereof) capturing it, allows a more nuanced examination of the two models. The inclusion of decay, which essentially creates a learning-relevant choice kernel [5], [6], as it maintains learning only for repeatedly chosen features, seems to improve overall model fit but fails to account for the immediate transition at the "point of learning." Perhaps this lag is due to its retrospective nature: first learn about everything, then decay anything that is revealed by participants' choices to have been learned (by the mode) in error. Models incorporating prospective attention fare better at accounting for behavior around this change-point, but rapidly lose their advantage as participants switch from exploration to exploitation. This effect may be due to the attention measures used diminishing in effectiveness, participants deploying less selective attention once they have learned the correct representation, or both. This comparison points to two other potential shortcomings of these models. Neither model explicitly accounts for the change-point between exploration and exploitation, even though it appears to be a distinct marker of the cognitive strategy employed by participants solving this task. Recent evidence suggests that confidence might govern this transition [7], suggesting that a model incorporating separate exploration and exploitation strategies may fare better at predicting the behavioral data. Of course, the transition may not be as abrupt as we have portrayed it to be, which such a model could attempt to capture. Here, reaction times for different choices may provide useful data that has been previously underexplored in the context of this task.

Another interesting finding is that the two models, which we have cast as conceptually different, explain the choice data equally well before the point of learning, that is, during the actual representation learning stage. Three possible explanations for this finding come to mind: 1) prospective and retrospective attention may contribute equally to the representation learning process itself, 2) the models may be disguising as one another in the learning phase, and 3) the differential contributions of the two processes are perhaps not separable with our task. In some sense, we cannot know if both models account for the cognitive strategy participants take, or possibly neither model captures it well.

Finally, from the perspective of model comparison methodology, our results reaffirm the value of diving beyond average model comparison metrics, such as mean cross-validated likelihood and BIC, and employing more granular comparisons. Examining specific cases in which competing models make different predictions enables more nuanced investigation than the overall goodness of fit of each model. Had we known in advance that such a stark behavioral change-point exists in the data, we could have simulated data using this "qualitative signature," which we expect a good model for the task to be able to recover, and evaluated the models accordingly [8].

6 References

[1] Y. Niv, R. Daniel, A. Geana, S. J. Gershman, Y. C. Leong, A. Radulescu, and R. C. Wilson, "Reinforcement learning in multidimensional environments relies on attention mechanisms," *The Journal of Neuroscience*, vol. 35, no. 21, pp. 8145–8157, May 27, 2015.

[2] Y. C. Leong, A. Radulescu, R. Daniel, V. DeWoskin, and Y. Niv, "Dynamic interaction between reinforcement learning and attention in multidimensional environments," *Neuron*, vol. 93, no. 2, pp. 451–463, Jan. 2017.

[3] N. J. Mackintosh, "A theory of attention: Variations in the associability of stimuli with reinforcement," *Psychological Review*, vol. 82, no. 4, pp. 276–298, 1975.

[4] J. Gottlieb, "Attention, learning, and the value of information," *Neuron*, vol. 76, no. 2, pp. 281–295, Oct. 18, 2012.

[5] B. Lau and P. W. Glimcher, "Dynamic response-by-response models of matching behavior in rhesus monkeys," *Journal of the Experimental Analysis of Behavior*, vol. 84, no. 3, pp. 555–579, Nov. 2005.

[6] R. Akaishi, K. Umeda, A. Nagase, and K. Sakai, "Autonomous mechanism of internal choice estimate underlies decision inertia," *Neuron*, vol. 81, pp. 195–206, Jan. 8, 2014.

[7] A. Boldt, C. Blundell, and B. De Martino, "Confidence modulates exploration and exploitation in value-based learning," 2017.

[8] R. C. Wilson and A. Collins, "Ten simple rules for the computational modeling of behavioral data," 2019.

Predicting Periodicity with Temporal Difference Learning

Kristopher De Asis
Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8
kldeasis@ualberta.ca

Brendan Bennett
Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8
babennet@ualberta.ca

Richard S. Sutton
Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8
rsutton@ualberta.ca

Abstract

Temporal difference (TD) learning is an important approach in reinforcement learning, as it combines ideas from dynamic programming and Monte Carlo methods in a way that allows for online and incremental model-free learning. A key idea of TD learning is that it is learning predictive knowledge about the environment in the form of value functions, from which it can derive its behavior to address long-term sequential decision making problems. The agent's horizon of interest, that is, how immediate or long-term a TD learning agent predicts into the future, is adjusted through a discount rate parameter. In this paper, we introduce an alternative view on the discount rate, with insight from digital signal processing, to include complex-valued discounting. Our results show that setting the discount rate to appropriately chosen complex numbers allows for online and incremental estimation of the Discrete Fourier Transform (DFT) of a signal of interest with TD learning. We thereby extend the types of knowledge representable by value functions, which we show are particularly useful for identifying periodic effects in the reward sequence.

Keywords: Reinforcement Learning, Online Learning, Signal Processing, Discrete Fourier Transform

Acknowledgements

The authors thank Roshan Shariff for insights and discussions contributing to the results presented in this paper, and the entire Reinforcement Learning and Artificial Intelligence research group for providing the environment to nurture and support this research. We gratefully acknowledge funding from Alberta Innovates – Technology Futures, the Alberta Machine Intelligence Institute, Google Deepmind, and from the Natural Sciences and Engineering Research Council of Canada.

1 Temporal Difference Learning

Temporal-difference (TD) methods [7] are an important approach in *reinforcement learning* as they combine ideas from dynamic programming and Monte Carlo methods. TD allows learning to occur from raw experience in the absence of a model of the environment's dynamics, like with Monte Carlo methods, while computing estimates which bootstrap from other estimates, like with dynamic programming. This provides a way for an agent to learn online and incrementally in both long-term prediction and sequential decision-making problems.

A key view of TD learning is that it is learning testable, predictive knowledge of the environment [11]. The learned value functions represent answers to predictive questions about how a signal will accumulate over time, given a way of behaving in the environment. A TD learning agent can continually compare its predictions to the actual outcomes, and incrementally adjust its world knowledge accordingly. In control problems, this signal is the reward sequence, and the value function represents the long-term cumulative reward an agent expects to receive when behaving greedily with respect to its current predictions about this signal.

A TD learning agent's time horizon of interest, or how long-term it is to predict into the future, is specified through a *discount rate* [10]. This parameter adjusts the weighting given to later outcomes in the sum of a sequence over time, trading off between only considering immediate or near-term outcomes and estimating the sum of arbitrarily long sequences. From this interpretation of its purpose, along with convergence considerations, the discount rate is restricted to be $\gamma \in [0, 1]$ in episodic problems, and $\gamma \in [0, 1)$ in continuing problems.

In this paper, we investigate whether meaningful information can be learned from relaxing the range of values the discount rate can be set to. In particular, we allow it to take on complex values, and instead restrict the magnitude of the discount rate, $|\gamma|$, to fall within the aforementioned ranges.

2 One-step TD and the MDP Formalism

The sequential decision-making problem in reinforcement learning is often modeled as a *Markov decision process* (MDP). Under the MDP framework, an *agent* interacts with an environment over a sequence of discrete time steps. At each time step t, the agent receives information about the environment's current *state*, $S_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states in the MDP. The agent is to use this state information to select an *action*, $A_t \in \mathcal{A}(S_t)$, where $\mathcal{A}(s)$ is the set of possible actions in state s. Based on the environment's current state and the agent's selected action, the agent receives a *reward*, $R_{t+1} \in \mathbb{R}$, and gets information about the environment's next state, $S_{t+1} \in \mathcal{S}$, according to the *environment model*: $p(s', r|s, a) = P(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$.

The agent selects actions according to a *policy*, $\pi(s, a) = P(A_t = a|S_t = s)$, which gives a probability distribution across actions $a \in \mathcal{A}(s)$ for a given state s, and is interested in the expected discounted return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \tag{1}$$

given a discount rate $\gamma \in [0, 1]$ and T equal to the final time step in an episodic task, or $\gamma \in [0, 1)$ and T equal to infinity for a continuing task.

Value-based methods approach the sequential decision-making problem by computing *value functions*, which provide estimates of what the return will be from a particular state onwards. In prediction problems, also referred to as *policy evaluation*, the goal is to estimate the return under a particular policy as accurately as possible, and a *state-value function* is often estimated. It is defined to be the expected return when starting in state s and following policy π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{2}$$

TD methods learn an approximate value function, such as $V \approx v_\pi$ for state-values, by computing an estimate of the return, $\hat{G}_t$. One-step TD methods compute $\hat{G}_t$ by taking an action in the environment according to a policy, sampling the immediate reward, and bootstrapping off of its current estimated value of the next state for the remainder of the return. The difference between this *TD target* and the value of the previous state is then computed, and the previous state's value is updated by taking a step proportional to this difference with a step size $\alpha \in (0, 1]$:

$$\hat{G}_t = R_{t+1} + \gamma V(S_{t+1}) \tag{3}$$

$$V(S_t) \leftarrow V(S_t) + \alpha[\hat{G}_t - V(S_t)] \tag{4}$$

3 Complex Discounting and the Discrete Fourier Transform

The discount rate has an interpretation of specifying the horizon of interest for the return, trading off between focusing on immediate rewards and considering the sum of longer sequences of rewards. It can also be interpreted as a *soft*

termination of the return [11, 8, 6], where an agent includes the next reward with probability γ, and terminates with probability $1-\gamma$, receiving a terminal reward of 0. From these interpretations, it is intuitive for the discount rate to fall in the range of $\gamma \in [0, 1)$ with the exception of episodic problems, where γ can be equal to 1.

With considerations for convergence, assuming the rewards are bounded, restricting the discount rate to be in this range makes the infinite sum (in the continuing case) of Equation 1 finite. However, this sum will remain finite when the magnitude of the discount rate is restricted to be $|\gamma| \in [0, 1)$, allowing for the use of negative discount rates up to -1, as well as complex discount rates within the complex unit circle.

While the use of alternative discount rates may result in some corresponding value function, a question arises regarding whether these values are meaningful, or if there is any situation in which an agent would benefit from this knowledge. First, we consider the implications of exponentiating a complex discount rate. We look at the exponential form of a complex number with magnitude A, and note that it can be expressed as a sum of sinusoids by Euler's Formula:

$$Ae^{-i\omega} = A\cos(\omega) - iA\sin(\omega) \tag{5}$$

From this, it is evident that exponentiating a complex number to the power of n corresponds to taking n steps around the complex unit circle with an angle of ω. Using this as a discount rate, we get the following return for some angle ω:

$$G_t^\omega = \sum_{k=0}^{T-t-1} e^{-i\omega k} A^k R_{t+k+1} \tag{6}$$

Complex discount rates weight a discounted reward sequence (with discount rate A) with two sinusoids, one along the real axis and one along the imaginary axis. This checks the cross correlation between a reward sequence and a sinusoid oscillating with a frequency of $\omega \frac{\text{rad}}{\text{step}}$ over an exponentially decaying window determined by A, allowing a TD learning agent to identify periodicity in the reward sequence at specified frequencies online and incrementally. While discounting by the magnitude A can distort the reward sequence, it primarily affects low frequencies which are unable to complete an oscillation within a discount rate's effective horizon.

Weighting the reward sequence with exponentiated complex numbers is equivalent to performing the Discrete Fourier Transform (DFT) from digital signal processing literature [2]. The DFT corresponds to the discrete form of the coefficients of a Fourier series, and thus each complex number encodes the amplitude and phase of a particular sinusoidal component of a sequence. Specifically, the normalized magnitude, $|G_t^\omega|/N$, corresponds to the amplitude, and the angle between the imaginary and real components, $\angle G_t^\omega$, gives the phase. The DFT is also an invertible, linear transformation [2]. Knowing the sequence length, and the sampling frequency (denoted f_s), one can invert the transform through a sum of sinusoids:

$$x_n = \sum_{k=0}^{N-1} \frac{|X_k|}{N} \cos\left(2\pi \frac{k}{N} f_s n + \angle X_k\right) \tag{7}$$

The learned approximate values of a TD agent represents the DFT of the return from a given state onward, and allows for identification of the signal's frequency information. However, the expected length of the sequence is typically not known by the agent, resulting in unnormalized amplitude information.

4 Experiments

In this section, we detail several experiments involving TD learning agents using complex-valued discounting.

4.1 Checkered Grid World

The *checkered grid world* environment is a 5×5 grid world with two terminal states on opposite corners. It has deterministic 4-directional movement, and moving into a wall keeps the agent in place. The agent starts in the center, and the board is colored with a checkered pattern which represents the reward distribution. One color represents a reward of 1 upon entry, and a reward of -1 for the other. Reaching a terminal state ends the episode with a reward of 11. Given that complex discounting computes the DFT, we would like to see whether an agent using complex discount rates can identify the structure of the reward distribution. We would also like to assess how well the expected reward sequence can be reconstructed through Equation 7, with knowledge of the expected sequence length.

This environment was treated as an on-policy policy evaluation task with no discounting ($|\gamma| = 1$). The agent behaved under an equiprobable-random behavior policy, and learned value functions corresponding to 114 equally spaced frequencies in the range $\omega \in [0, 2\pi)$. Expected Sarsa [13] was used, and state-values were computed from the action-values.

We performed 100 runs, and the value of the starting state, represented by the complex number's magnitude and phase information, was plotted for each frequency after the 250th episode. The resulting learned DFT of the starting state can be

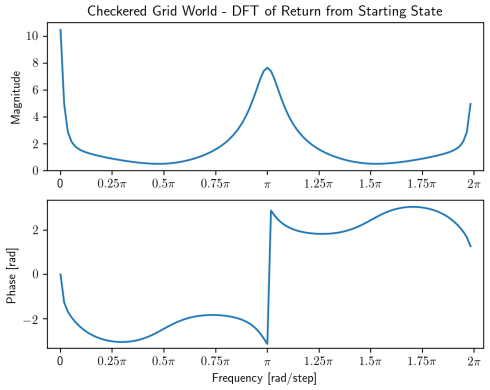
seen in Figure 1. Of note, the frequencies are relative to the agent's sampling frequency, with $\omega = 2\pi$ being 1 sample per step. In the learned DFT, the magnitude of the value at $\omega = 0$ gives what a standard TD agent with $\gamma = e^{-i0} = 1$ would have learned. There is a relatively large magnitude at half the agent's sampling frequency, meaning the agent has large confidence in the existence of an oscillation at a rate of half a cycle per time step. This corresponds to the environment's rewards alternating between 1 and -1, as this pattern takes two time steps to complete a cycle.

Next, we reconstructed the expected reward sequence. Knowing the expected episode length, we use the learned values of 38 equally spaced frequencies in $[0, 2\pi)$, and evaluated Equation 7 for 38 steps. The reconstruction can be seen in Figure 2. Qualitatively, the reconstructed sequence captures several aspects of the structure of the return. The oscillations in the sequence have a period of 2, and begin with an approximate amplitude of 1. It has a positive mean from the positive reward upon termination, and its sum gives the correct standard undiscounted return. Of note, the sequence appears to decay, and doesn't end with a reward of 11. This is because when the agent bumps a wall, the periodic pattern shifts. The earliest this can occur is in 3 steps, approximately where the exponential decay begins in the reconstruction.

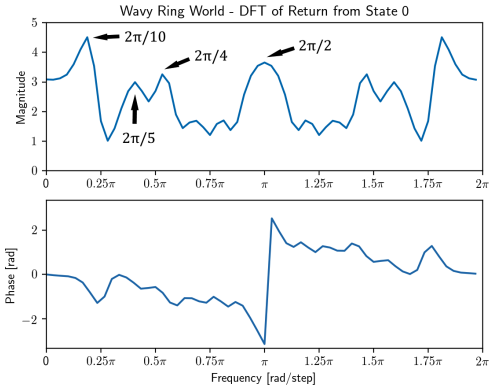
4.2 Wavy Ring World

To assess the idea in a continuing setting with function approximation, we designed the *wavy ring world* environment. This environment has 20 states arranged in a ring. The agent starts in state 0 and traverses the states in a fixed direction. Binary feature vectors to be used with linear function approximation were produced with tile coding [9]. Specifically, we used 6 overlapping tilings and each tile spanned 1/3-rd of the 20 states. The reward for leaving a state $s \in \{0, 1, 2, ..., 19\}$, $R(s)$, consisted of the sum of four state-dependent, unit-amplitude sinusoids with periods of 2, 4, 5, and 10 states.

A TD agent learned complex-valued weights for each of 64 equally spaced frequencies in the range $\omega \in [0, 2\pi)$, and the magnitude of each discount rate was $|\gamma| = 0.9$. We extracted the state values from the learned weights, and the resulting DFT of the return from state 0 can be seen in Figure 1. In the learned DFT, we can see that it still has relatively large peaks at various frequencies in the magnitude plot. Looking at the frequencies at which these peaks occur, they correspond to the frequencies of the reward function $R(s)$. One might be concerned that the peaks are not equal, because the reward signal's sinusoids have identical amplitudes. However, discounting distorts the signal, and the analytic DFT has matching, unequal peaks. To illustrate this, we evaluate Equation 7 to invert the learned DFT (knowing the reward signal's period), and un-discount the sequence. The reconstructed reward sequence from state 0 can be seen in Figure 2.



(a) Checkered grid world results



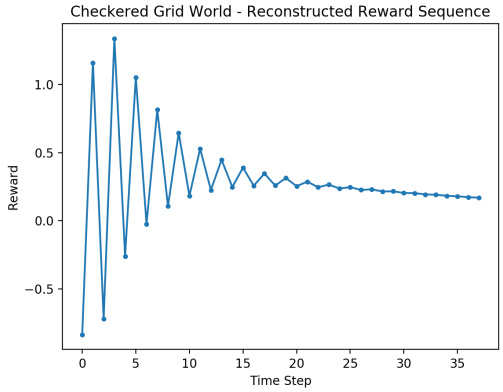
(b) Wavy ring world results

Figure 1: Learned DFTs for each environment. Note that they are symmetric about half of the sampling frequency. This frequency is referred to as the *Nyquist frequency* [2], which, due to aliasing, is the largest detectable frequency.

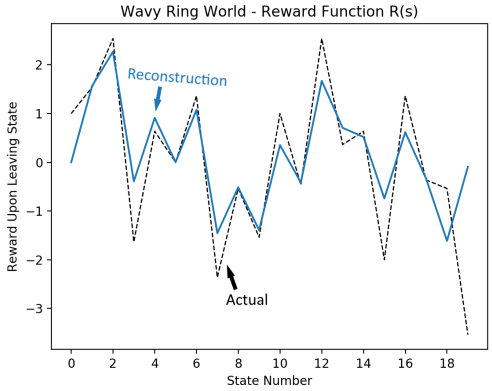
5 Discussion and Conclusions

In our experiments, we showed that a TD agent using complex discount rates can identify periodic patterns in the return. This is due to complex discount rates allowing for incremental estimation of the return's DFT. We also showcased a way of inverting the DFT, with knowledge of the sequence length, allowing for reconstructing the expected reward sequence.

We focused on a case where periodicity exists in the environment. This may have implications for problems with sound or image data, as the DFT is typically used as a post-processing tool in those applications. In general, this approach identifies *policy-contingent* frequency information. An agent behaving under a policy which led it in circles would induce similar alternating patterns in the reward sequence. For example, using a robot's joint position as a time step's reward, robot locomotion would induce periodicity in the reward signal. Awareness of periodicity may have implications in the



(a) Checkered grid world reward sequence reconstruction



(b) Wavy ring world reward sequence reconstruction

Figure 2: Reward sequence reconstructions for both environments from inverting the learned DFT.

options framework [12], as it may offer insight regarding where an option should terminate. It may also have use in exploration, where if state features are used as rewards, an agent avoiding periodicity may seek out novel states.

Beyond identifying periodicity, being able to roughly reconstruct the expected reward sequence, an agent might be able to base decisions on properties like reward sparsity, or noise in the reward. Reconstructing the sequence can be seen as recovering information lost from computing the sum of the rewards, which is different but comparable to distributional reinforcement learning [1, 3] which recovers information lost from computing the sum's expectation.

Prior work used a Fourier basis as a state representation in reinforcement learning [5], that complex discounting may allow for incremental estimation of a similar representation. Also, in the deep reinforcement learning setting, learning about many frequencies in parallel may have the representation learning benefit of predicting auxiliary tasks [4].

References

[1] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 2017.

[2] E. O. Brigham. *The Fast Fourier Transform and Its Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[3] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. *CoRR*, abs/1710.10044, 2017.

[4] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *CoRR*, abs/1611.05397, 2016.

[5] G. D. Konidaris, S. Osentoski, and P. S. Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, pages 380–385, 2011.

[6] J. Modayil, A. White, and R. S. Sutton. Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behaviour*, 22(2):146–160, 2014.

[7] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[8] R. S. Sutton. TD model: Modeling the world at a mixture of time scales. Technical report, Amherst, MA, USA, 1995.

[9] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, 1996.

[10] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd edition, 2018. Manuscript in preparation.

[11] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *AAMAS*, pages 761–768. IFAAMAS, 2011.

[12] R. S. Sutton, D. Precup, and S. P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[13] H. van Seijen, H. van Hasselt, S. Whiteson, and M. Wiering. A theoretical and empirical analysis of Expected Sarsa. In *Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184, 2009.

A Bayesian Approach to Robust Reinforcement Learning

Esther Derman
Technion, Israel
estherderman@technion.ac.il

Daniel J. Mankowitz
Deepmind, UK
dmankowitz@google.com

Timothy A. Mann
Deepmind, UK
timothymann@google.com

Shie Mannor
Technion, Israel
shie@ee.technion.ac.il

Abstract

In sequential decision making problems, Robust Markov Decision Processes (RMDPs) intend to ensure robustness with respect to changing or adversarial system behavior. In this framework, transitions are modeled as arbitrary elements of a known and properly structured *uncertainty set* and a robust optimal policy can be derived under the worst-case scenario. However, in practice, the uncertainty set is unknown and must be constructed based on available data. Most existing approaches to robust reinforcement learning (RL) build the uncertainty set upon a fixed batch of data before solving the resulting planning problem. Since the agent does not change its uncertainty set despite new observations, it may be overly conservative by not taking advantage of more favorable scenarios. Another drawback of these approaches is that building the uncertainty set is computationally inefficient, which prevents scaling up online learning of robust policies. In this study, we address the issue of learning in RMDPs using a Bayesian approach. We introduce the Uncertainty Robust Bellman Equation (URBE) which encourages exploration for adapting the uncertainty set to new observations while preserving robustness. We propose a URBE-based algorithm, DQN-URBE, that scales this method to higher dimensional domains. Our experiments show that the derived URBE-based strategy leads to a better trade-off between less conservative solutions and robustness in the presence of model misspecification. In addition, we show that the DQN-URBE algorithm can adapt significantly faster to changing dynamics online compared to existing robust techniques with fixed uncertainty sets.

Keywords: Robust Markov Decision Processes, Online Learning, Deep Reinforcement Learning, Deep Q-Network

Acknowledgements

The authors would like to thank Chen Tessler for his help and useful comments on this paper.

1 Introduction

Markov Decisions Processes (MDPs) are used for solving sequential decision making problems with varying degrees of uncertainty. Two types of uncertainty can be encountered: the internal uncertainty due to the stochasticity of the system and the uncertainty in the transition and reward parameters [13]. In order to mitigate the second type of uncertainty, the Robust-MDP (RMDP) framework considers the unknown parameters to be a member of a known uncertainty set [14, 7, 21]. An optimal solution to the robust RL problem then corresponds to the strategy that maximizes the worst-case performance and it can be derived using dynamic programming [7, 19]. However, planning in RMDPs can lead to overly conservative solutions. This is due to two main reasons. Firstly, the uncertainty set has to be *rectangular* in order for the problem to be computationally tractable, which means that it must be structured as independent sets of MDP models for each state [21]. Attempts to circumvent rectangular sets in RMDPs include the works [11, 12, 20, 6]. Secondly, the difficulty of constructing uncertainty sets can result in too large sets and consequently lead to overly-pessimistic strategies [18].

In this work, we introduce a Bayesian framework for robust RL and address the first Bayesian algorithm that (1) accounts for changing dynamics online (2) tackles conservativeness thanks to a variance bonus that detects changes in the level of adversity. This variance is proven to satisfy an Uncertainty Robust Bellman Equation (URBE), that is estimated using dynamic programming. Besides being scalable to complex domains, our approach leads to less conservative results than existing planning methods for RMDPs while ensuring robustness to model misspecification. Our specific contributions are: (1) An Uncertainty Robust Bellman Equation (URBE) that encourages robust exploration and prevents overly conservative solutions. It also adapts the uncertainty set online to new observations allowing the agent to deal with sudden changes to the underlying environment dynamics.; (2) A scalable algorithm, DQN-URBE, that utilizes URBE to learn less conservative solutions that are still robust to model misspecification; (3) Experimental evidence in three domains illustrating an improved trade-off between overly, conservative robust behaviour and less, conservative, improved performance for the resulting DQN-URBE policy.

Related Work Proposals for learning an uncertainty set in a data-driven manner have rarely been addressed in RL literature. Russel & Petrik [18] designed a robustification procedure that enlarges an initially trivial uncertainty set in a safe manner. Although it leads to tighter uncertainty sets, their algorithm proceeds offline with a fixed batch of data and is not scalable due to its high computational complexity. Previous work [8, 9] has used the "optimism in face of uncertainty" principle to detect adversarial state-action pairs online and compute an optimistic minimax policy accordingly. Although these methods have been proven to be statistically efficient, they require an exhaustive computation for each state-action pair. This leads to solutions that are intractable for all but small problems. Previous work [16] has stressed the advantages of posterior sampling methods over existing algorithms driven by optimism. However, most of the existing work on posterior sampling methods studied finite tabular MDPs. The Uncertainty Bellman Equation (UBE) work [15] addressed this shortcoming and proposed an online algorithm that scales naturally to large domains. Their method learns the posterior variance of the value for guiding exploration when the true dynamics of the MDP are unknown. Yet, their approach does not deal with adversarial transitions.

2 Problem Formulation

We consider an RMDP $\langle \mathcal{S}, \mathcal{A}, r, \mathcal{P} \rangle$ of finite state and action spaces, where each episode has finite horizon length $H \in \mathbb{N}$. At step h, an agent is in state s^h, selects an action a^h according to a stochastic policy $\pi^h : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{A}}$ that maps each state to a probability distribution over the action space, $\Delta_{\mathcal{A}}$ denoting the set of distributions over $\mathcal{A}$. The agent then gets a deterministic reward r^h and transitions to state s^{h+1} according to an arbitrary transition $p_{s^h,a^h} \in \mathcal{P}_{s^h,a^h} \subseteq \Delta_{\mathcal{S}}$. This induces a rectangular structure on the uncertainty set, which is formally defined in the following.

Definition 2.1 (Rectangularity)**.** *Given an RMDP $\langle \mathcal{S}, \mathcal{A}, r, \mathcal{P} \rangle$, the uncertainty set $\mathcal{P}$ is said to be (s,a)-rectangular if $\mathcal{P} := \bigotimes_{s \in \mathcal{S}, a \in \mathcal{A}} \mathcal{P}_{s,a}$, where $\mathcal{P}_{s,a}$ are subsets of the probability simplex $\Delta_{\mathcal{S}}$.*

The robust Q-value at step h, state s, action a and under policy $\pi := (\pi^1, \cdots, \pi^H)$ is the expected total return under the worst-case scenario resulting from taking action a at s and following policy π thereafter: $Q^h_{sa} := \inf_{p \in \mathcal{P}} \mathbb{E}\left[\sum_{l=h}^{H} r^l \mid s^h = s, a^h = a, \pi, p\right]$. When it is clear from the context, we suppress the dependence on π for notational convenience. A robust optimal policy is derived by maximizing the expected total worst-case return: $J(\pi) := \inf_{p \in \mathcal{P}} \mathbb{E}^{\pi,p}\left[\sum_{h=1}^{H} r^h\right]$. Assuming a rectangular structure on $\mathcal{P}$, the robust Bellman operator $\mathcal{T}^h$ for policy π at step h relates the robust value at h to the robust value at following steps [7, 14]: $\mathcal{T}^h Q^{h+1}_{sa} = r^h_{sa} + \inf_{p \in \mathcal{P}} \sum_{s' \in \mathcal{S}, a' \in \mathcal{A}} \pi^h_{s'a'} p^h_{sas'} Q^{h+1}_{s'a'}$.

3 The Uncertainty Robust Bellman Equation

Posterior uncertainty sets Define ϕ_p as a prior distribution according to which state transitions are generated. Assume furthermore that ϕ_p is a product of $|\mathcal{S}| \cdot |\mathcal{A}|$ independent Dirichlet priors on each distribution p_{sa} over next states, that

is $\phi_p = \prod_{s,a} \phi_{sa}$, where ϕ_{sa} is Dirichlet. Given an observation history $\mathcal{H} = \langle (s_1, a_1), (s_2, a_2), \ldots, (s_h, a_h) \rangle \in (\mathcal{S} \times \mathcal{A})^h$ induced by a policy π and a confidence level $\psi_{sa} \in \mathbb{R}^+$ for each state-action pair, we can construct a subset of transition probabilities: $\widehat{\mathcal{P}}^h_{sa}(\psi_{sa}) = \{p_{sa} \in \Delta_{\mathcal{S}} : \|p_{sa} - \bar{p}_{sa}\|_1 \leq \psi_{sa}\}$ where $\bar{p}_{sa}$ is the nominal transition given by $\bar{p}_{sa} = \mathbb{E}[p_{sa} \mid \mathcal{H}]$. This construction forms a rectangular uncertainty set $\widehat{\mathcal{P}}^h(\psi) := \bigotimes_{s,a} \widehat{\mathcal{P}}^h_{sa}(\psi_{sa})$. We call it a *posterior uncertainty set* and will omit the dependence in ψ for ease of notation.

Posterior over robust Q-values The simulation proceeds as follows: at each episode t, we sample a transition matrix according to ϕ_p. For a fixed policy π, we collect observation history. We then construct a posterior uncertainty set based on observed data. A posterior over robust Q-values can then be obtained via $\widehat{Q}^h_{sa} = r^h_{sa} + \inf_{p \in \widehat{\mathcal{P}}^h_{sa}} \sum_{s',a'} \pi^h_{s'a'} p_{sas'} \widehat{Q}^{h+1}_{s'a'}$, with $\widehat{Q}^{H+1}_{sa} = 0$. A worst-case transition at step h is then defined as

$$\widehat{p}^h_{sa} \in \arg \inf_{p \in \widehat{\mathcal{P}}^h_{sa}} \sum_{s',a'} \pi^h_{s'a'} p_{sas'} \widehat{Q}^{h+1}_{s'a'} \tag{1}$$

Posterior variance of robust Q-values For the regular MDP setting, O'Donogue et al. [15] showed that the conditional variance of posterior Q-values can be bounded by a quantity that satisfies a Bellman recursion formula. In Bayesian robust RL, a similar upper bound can be derived, as stated below. We first introduce our notation as well as common assumptions:

Notation 3.1. *Define $\mathcal{F}_t$ as a minimal sigma-algebra that contains all of the available information up to episode t. Denote by $\mathbb{E}_t[X]$ the expectation of random variable X conditioned on $\mathcal{F}_t$. Similarly, the conditional variance is $\mathbf{var}_t X := \mathbb{E}_t \left[(X - \mathbb{E}_t[X])^2 \right]$.*

Assumption 3.1. *For any episode, the graph resulting from a worst-case transition model is directed and acyclic. Furthermore, for all $(s, a) \in \mathcal{S} \times \mathcal{A}$, the rewards are bounded: $-R_{\max} \leq r_{sa} \leq R_{\max}$. It follows that for all s, a and h: $| Q^h_{sa} | \leq H R_{\max} =: Q_{\max}$.*

Theorem 3.1 (Solution of URBE)**.** *For any worst-case transition $\widehat{p}$ as defined in equation (1) and any policy π, under Assumption 3.1, there exists a unique mapping w that satisfies the uncertainty robust Bellman equation: $w^h_{sa} = \nu^h_{sa} + \sum_{s' \in \mathcal{S}, a' \in \mathcal{A}} \pi^h_{s'a'} \mathbb{E}_t(\widehat{p}^h_{sas'}) w^{h+1}_{s'a'}$, for all $(s, a) \in \mathcal{S} \times \mathcal{A}$ and $h = 1, \cdots, H$ where $w^{H+1} = 0$ and $\nu^h_{sa} := Q^2_{\max} \sum_{s' \in \mathcal{S}} \frac{\mathbf{var}_t \widehat{p}^h_{sas'}}{\mathbb{E}_t \widehat{p}^h_{sas'}}$. Furthermore, $w \geq \mathbf{var}_t \widehat{Q}$.*

To prove this result, we first bound the posterior variance of the robust Q-value. Existence and uniqueness of URBE is then established using robust dynamic programming (Exercise 1.5 in [2]).

A classical difficulty in Bayesian approaches is to compute the posterior distribution. The Bayesian central limit theorem (Result 8 in [1]) ensures that under smoothness assumptions on prior and likelihood functions, the posterior distribution converges to a Gaussian as the size of the data set increases. We thus approximate the posterior over robust Q-values as $\mathcal{N}(\bar{Q}, \mathbf{diag}(w))$, where w is the solution to URBE and $\bar{Q}$ is the unique solution to $\bar{Q}^h_{sa} = r^h_{sa} + \sum_{s',a'} \pi^h_{s'a'} \mathbb{E}_t(\widehat{p}_{sas'})^h \bar{Q}^{h+1}_{s'a'}$ for $h = 1, \cdots, H$ and $\bar{Q}^{H+1} = 0$, with $\widehat{p}^h_{sa} \in \arg\inf_{p \in \widehat{\mathcal{P}}^h_{sa}} \sum_{s',a'} \pi^h_{s'a'} p_{sas'} \bar{Q}^{h+1}_{s'a'}$.

Theorem 3.1 reveals another quantity ν that only depends on local state and action pairs. We call it the *robust local uncertainty*. Similarly to the non-robust setup, the robust local uncertainty can be modeled as a positive constant divided by the visit counts $n^h_{sa} := (n^h_{sas'})_{s' \in \mathcal{S}}$ at step h, up to episode t. In the case of linear approximation for the Q-value estimate, O'Donogue et al. introduced an estimate for the visit counts [15]. Likewise, defining $\widehat{Q}^h_{sa} := \phi_s^T \theta_a$ with $\phi : \mathcal{S} \to \mathbb{R}^d$ designating state features and θ_a being weights parameters that need to be learned (one for each action), we have $(\widehat{n}^h_{sa})^{-1} = \phi_s^T (\Phi_a^T \Phi_a)^{-1} \phi_s$, where Φ_a is the matrix of ϕ_s-s stacked row-wise with action a being taken at s. We can then estimate the robust local uncertainty as $\widehat{\nu}^h_{sa} = \beta^2 \phi_s^T (\Phi_a^T \Phi_a)^{-1} \phi_s$ [15]. As it receives a new sample ϕ, the agent needs to update the matrix $\Sigma_a := (\Phi_a^T \Phi_a)^{-1}$ by affecting to it the following Sherman-Morrison-Woodbury formula [5]: $\Sigma_a^+ := \Sigma_a - (\Sigma_a \phi \phi^T \Sigma_a)/(1 + \phi^T \Sigma_a \phi)$. The neural network representation proceeds similarly, provided that we treat all layers as feature extractors and finally apply a linear activation function. In that case, we still have $\widehat{Q}^h_{sa} = \phi_s^T \theta_a$, where ϕ_s is the output network up to last layer for state s and θ_a are the parameters of the last layer for action a. We will use this technique in Algorithm 1.

4 DQN-URBE Algorithm

The robust Bellman equation utilizes a robust TD-error as a loss criterion for learning a minimax policy [17]. The robust TD error to be minimized is defined as: $\delta^h := r(s^h, a^h) + \gamma \inf_{p \in \mathcal{P}} \sum_{s' \in \mathcal{S}} p(s^h, a^h, s') \max_{a' \in \mathcal{A}} Q(s', a') - Q(s^h, a^h)$, where the uncertainty set is fixed. This method has been shown to lead to robust yet overly conservative behavior [10, 3].

In order to generate a less conservative solution, we present our DQN-URBE algorithm (Algorithm 1). Since finding a solution to URBE requires solving a robust optimization problem at each episode, it is computationally costly and not

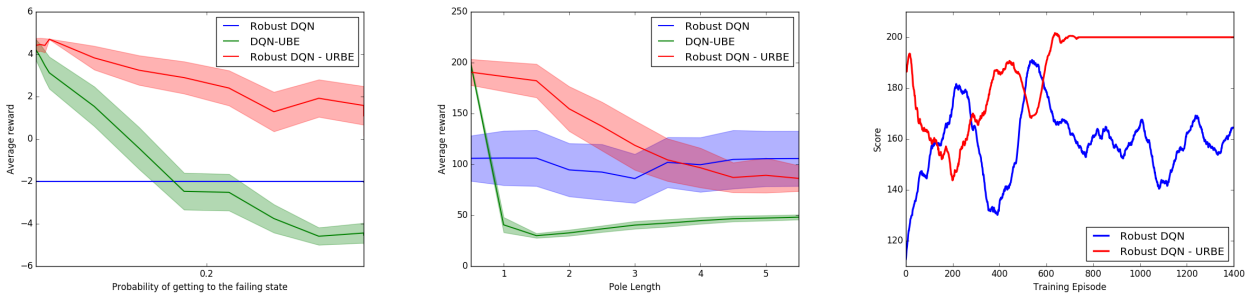
scalable. We avoid this problem by keeping the uncertainty set fixed and finite, and add the robust local uncertainty as an exploration bonus. In practice, DQN-URBE consists of a neural network architecture that has two output heads: one attempts to learn the optimal robust Q-function of a fixed uncertainty set via the robust-DQN subroutine as described in [10]; the other attempts to estimate the robust uncertainty for the robust Q-function, as mentioned in Section 3. We added stop-gradients to prevent the posterior variance from affecting the robust Q-network parameters and vice-versa.

Algorithm 1 DQN - URBE

Input: Neural network for robust Q and w estimates; Robust DQN subroutine `robustDQN`; Hyperparameter $\beta > 0$
Initialize: $\Sigma_a = \mu \cdot I$ for $a \in \mathcal{A}$ with $\mu > 0$; Initial state and action $(s, a) \in \mathcal{S} \times \mathcal{A}$
for $t = 1, \cdots$ **do**
  for $h = 2$ **to** $H + 1$ **do**
    Retrieve $\phi(s)$ from robust Q-network
    Observe s' and receive reward r
    Compute $\widehat{Q}^h_{s'b}$ and $w^h_{s'b}$ for all action b
    Sample $\zeta_b \sim \mathcal{N}(0, 1)$ for all b and compute:

$$a' = \arg\max_b \left(\widehat{Q}^h_{s'b} + \beta \zeta_b \sqrt{w^h_{s'b}} \right) \text{ and } y = \begin{cases} \phi(s)^T \Sigma_a \phi(s) & \text{if } h = H + 1 \\ \phi(s)^T \Sigma_a \phi(s) + \gamma^2 w^h_{s'a'} & \text{otherwise} \end{cases}$$

    Take gradient step on w w.r.t. loss $(y - w^{h-1}_{sa})^2$
    Update robust Q-values using `robustDQN`
    Update Σ_a according to the Sherman-Morrison-Woodbury formula: $\Sigma_a^+ := \Sigma_a - (\Sigma_a \phi \phi^T \Sigma_a)/(1 + \phi^T \Sigma_a \phi)$.
    Take action a'
    $s \leftarrow s', a \leftarrow a'$
  end for
end for



(a) (b) (c)

Figure 1: (a) Mar's Rover domain - 10x10 grid. (b) Cartpole: Testing rewards for DQN-URBE (c) Cartpole: Training score *after* changing the pole length. DQN-URBE is less conservative than robust DQN while it ensures robustness to changing dynamics

5 Experiments

We first checked the evolving performance of DQN-URBE across changing dynamics and executed it on a 10×10 grid-world domain. The agent starts at a random state from the top left of the grid and is required to travel to the goal located in the bottom right corner in order to get a high reward R_{success}. On each step, the agent can either be brought back to a terminating state with probability p and get a negative reward R_{fail} if it chose to move towards the goal. Otherwise, it moves into the chosen direction and receives a small negative reward R_{step}. DQN-UBE [15] and robust DQN [4] have been used as baselines. After training these three agents on a nominal probability of $p = 0.005$, we tested their robustness against model mispecification. Figure 1(a) shows that the robust agent is overly conservative since it keeps moving backwards, while the UBE agent's performance drops down under model mispecification as opposed to the URBE agent which takes adversarial transitions into account. We then tested DQN-URBE on Cartpole after a training of 4000 episodes, with 200 steps for each. Figure 1(b) shows that DQN-UBE is very sensitive to changing dynamics although it performs well on the nominal model. Robust DQN is too conservative despite its stability along different pole lengths. DQN-URBE leads to a strategy that performs well under the nominal model and is more stable than UBE under model mispecification. It also leads to a performance that is comparable to robust DQN when the pole length is high. In order to test the exploration

capacity of the robust agents, we also compared the sensitivity of robust DQN with DQN-URBE to changing dynamics during training. In practice, we waited for both agents to converge before changing the nominal length from 1 to 0.5. Figure 1(c) shows the training score after convergence of both agents. that URBE recovers faster than robust DQN besides reaching maximal reward.

6 Discussion

We presented a Bayesian approach to learning less conservative solutions when solving Robust MDPs. This is achieved using the Uncertainty Robust Bellman Equation (URBE), our adaptation of the UBE equation, which encourages safe exploration and implicitly modifies the uncertainty set online using new observations. We scale this approach to higher dimensional domain using the DQN-URBE algorithm and show the ability of the agent to learn less conservative solutions in a toy MDP, a Mar's rover domain and Open AI gym's Cartpole domain. Finally, we show the ability of the agent to adapt to changing dynamics significantly faster than a robust DQN agent during training. Our approach shed light on the advantages of adding a variance bonus to robust Q-learning for encouraging safe exploration in lowering the conservativeness of robust strategies. Further work should analyze the asymptotic behavior of our URBE-based method as well as the impact of the size of the posterior uncertainty set on the posterior variance of robust Q-values.

References

[1] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Science and Business Media, 2013.

[2] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 2 edition, 2000.

[3] Esther Derman, Daniel J. Mankowitz, and Timothy A. Mann. Soft-robust actor-critic policy-gradient. *UAI*, 2018.

[4] Shirli Di-Castro Shashua and Shie Mannor. Deep Robust Kalman Filter. *arXiv preprint arXiv:1703.02310v1*, 2017.

[5] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[6] Vineet Goyal and Julien Grand-Clement. Robust Markov decision process: Beyond rectangularity. *arXiv preprint arXiv:1811.00215v4*, 2019.

[7] Garud N. Iyengar. Robust Dynamic Programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

[8] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

[9] Shiau Hong Lim, Huan Xu, and Shie Mannor. Reinforcement learning in robust Markov decision processes. *Mathematic of Operations Research*, 41(4):1325–1353, November 2016.

[10] Daniel J Mankowitz, Timothy A Mann, Shie Mannor, Doina Precup, and Pierre-Luc Bacon. Learning Robust Options. In *AAAI*, 2018.

[11] Shie Mannor, Ofir Mebel, and Huan Xu. Lightning Does Not Strike Twice: Robust MDPs with Coupled Uncertainty. In *ICML*, 2012.

[12] Shie Mannor, Ofir Mebel, and Huan Xu. Robust MDPs with k-Rectangular Uncertainty. *Mathematics of Operations Research*, 41(4):1484–1509, 2016.

[13] Shie Mannor, Duncan Simester, Peng Sun, and John N. Tsitsiklis. Bias and Variance Approximation in Value Function Estimates. *Management Science*, 53(2):308–322, 2007.

[14] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):783–798, 2005.

[15] Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The Uncertainty Bellman Equation and Exploration. *ICML*, 2018.

[16] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? *ICML*, pages 2701–2710, 2017.

[17] Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under Model Mismatch. *31st Conference on Neural Information Processing Systems*, 2017.

[18] Reazul Hasan Russel and Marek Petrik. Tight bayesian ambiguity sets for robust MDPs. *Neural Information Processing Systems*, 2018.

[19] Aviv Tamar, Shie Mannor, and Huan Xu. Scaling up robust MDPs using function approximation. *ICML*, 32:1401–1415, 2014.

[20] Andrea Tirinzoni, Marek Petrik, Xiangli Chen, and Brian Ziebart. Policy-conditioned uncertainty sets for robust Markov decision processes. *Advances in Neural Information Processing Systems*, pages 8953–8963, 2018.

[21] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, February 2013.

Soft-Robust Actor-Critic Policy-Gradient

Esther Derman
Technion, Israel
estherderman@technion.ac.il

Daniel J. Mankowitz
Deepmind, UK
dmankowitz@google.com

Timothy A. Mann
Deepmind, UK
timothymann@google.com

Shie Mannor
Technion, Israel
shie@ee.technion.ac.il

Abstract

Robust reinforcement learning aims to derive an optimal behavior that accounts for model uncertainty in dynamical systems. However, previous studies have shown that by considering the worst-case scenario, robust policies can be overly conservative. Our *soft-robust* (SR) framework is an attempt to overcome this issue. In this paper, we present a novel Soft-Robust Actor-Critic algorithm (SR-AC). It learns an optimal policy with respect to a distribution over an uncertainty set and stays robust to model uncertainty but avoids the conservativeness of traditional robust strategies. We show the convergence of SR-AC and test the efficiency of our approach on different domains by comparing it against regular learning methods and their robust formulations.

Keywords: Reinforcement Learning, Robust Markov Decision Processes, Policy-Gradient

Acknowledgements

This work was partially funded by the Israel Science Foundation under contract 1380/16 and by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 306638 (SUPREL).

1 Introduction

Markov Decision Processes (MDPs) are commonly used to model sequential decision making in stochastic environments. A strategy that maximizes the expected accumulated reward is considered optimal and can be learned from sampling. However, besides the uncertainty that results from stochasticity of the environment, model parameters are often estimated from noisy data or can change during testing [11, 15]. This second type of uncertainty can significantly degrade the performance of model compared with the optimal strategy.

Robust MDPs were proposed to address this problem [6, 14, 18]. In this framework, a transition model is assumed to belong to a known uncertainty set and an optimal strategy is learned under the worst parameter realizations. Although the robust approach is computationally efficient when the uncertainty set is state-wise independent, compact and convex, it can lead to overly conservative results [19, 20, 9, 10].

For example, consider a business scenario where an agent's goal is to make as much money as possible. It can either create a startup which may make a fortune but may also result in bankruptcy. Alternatively, it can choose to live off school teaching and have almost no risk but low reward. By choosing the teaching strategy, the agent may be overly conservative and not account for opportunities to invest in his own promising projects. Our claim is that one could relax this conservativeness and construct a softer behavior that interpolates between being aggressive and robust. Ideally, the *soft-robust* agent should stay agnostic to unknown parameters that may affect the agent's income such as risky projects or low risk salary, but still be able to take advantage of the startup experience.

In this work, we focus on learning a *soft-robust policy* (SR policy) by incorporating soft-robustness into an online actor-critic algorithm and show its convergence properties. Our specific contributions are: (1) A SR derivation of the objective function for policy-gradient; (2) A SR-AC algorithm that uses stochastic approximation to learn a variant of distributionally robust policy in an online manner; (3) Convergence proofs of SR-AC; (4) An experiment with our framework in different domains that shows the efficiency of SR behaviors in a continuous action space as well. We refer the reader to [4] for an extended version of our work with theoretical proofs.

2 Background

In this section, we introduce the background material related to our SR approach.

Robust MDP A robust MDP is a tuple $\langle \mathcal{X}, \mathcal{A}, r, \mathcal{P} \rangle$ where $\mathcal{X}$ and $\mathcal{A}$ are finite state and action spaces respectively, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is a deterministic and bounded reward function and $\mathcal{P}$ is a set of transition matrices. We assume that $\mathcal{P}$ is structured as a cartesian product $\bigotimes_{x \in \mathcal{X}} \mathcal{P}_x$, which is known as the rectangularity assumption [14]. For $x, y \in \mathcal{X}$ and $a \in \mathcal{A}$, denote by $p(x, a, y)$ the probability of getting from state x to state y given action a. At timestep t, the agent is in state x_t and chooses an action a_t according to a stochastic policy $\pi : \mathcal{X} \to \mathcal{M}(\mathcal{A})$ that maps each state to a probability distribution over the action space, $\mathcal{M}(\mathcal{A})$ denoting the set of distributions over $\mathcal{A}$. It then gets a reward r_{t+1} and is brought to state x_{t+1} with probability $p(x_t, a_t, x_{t+1})$.

Policy-Gradient Methods A policy π is parameterized and estimated by optimizing an objective function using stochastic gradient descent [17]. Actor-critic methods attempt to reduce the variance of policy-gradient by using a critic that estimates the value function and helps evaluating the policy [5, 1]. The actor then uses this signal to update policy parameters in the gradient direction of the objective function [7, 2].

Deep Reinforcement Learning Algorithms In DQN [12, 13], a neural network approximates the Q-function. The agent is then trained by optimizing the induced TD loss function thanks to stochastic gradient descent. Since DQN acts greedily at each iteration, it can only handle small action spaces. Deep Deterministic Policy-Gradient (DDPG) is an *off-policy* algorithm that can learn behaviors in continuous action spaces [8]. It is based on an actor-critic architecture that follows the same baseline as DQN. The critic estimates the current Q-value of the actor using a TD-error while the actor is updated according to the critic. This update is based on the chain rule principle which establishes equivalence between the stochastic and the deterministic policy gradient [16].

3 Soft-Robust Policy-Gradient

Unlike robust MDPs that maximize the worst-case performance, we fix a prior on how transition models are distributed over the uncertainty set. A distribution over $\mathcal{P}$ is defined as a product measure $\omega := \bigotimes_{x \in \mathcal{X}} \omega_x$ [19, 20]. This defines a probability distribution ω_x over $\mathcal{P}_x$ independently for each state. Intuitively, ω can be thought as the way the adversary distributes over different transition models. The product structure then means that this adversarial distribution only depends on the current state without taking its whole trajectory into account.

We call *SR average reward* the SR objective $\bar{J}(\pi) := \mathbb{E}_{p \sim \omega}[J_p(\pi)]$, where $J_p(\pi)$ is the average reward under transition p, i.e. $J_p(\pi) = \lim_{T \to +\infty} \mathbb{E}^p[\frac{1}{T} \sum_{t=0}^{T-1} r_{t+1} \mid \pi]$. The *SR differential reward* is given by $\bar{Q}^\pi(x, a) := \mathbb{E}_{p \sim \omega}[Q_p^\pi(x, a)]$ where

$Q_p^\pi(x,a) := \mathbb{E}^p[\sum_{t=0}^{+\infty} r_{t+1} - J_p(\pi)|x_0 = x, a_0 = a, \pi]$. Similarly, the *SR value function* is $\bar{V}^\pi(x) := \sum_{a\in\mathcal{A}} \pi(x,a)\bar{Q}^\pi(x,a) = \mathbb{E}_{p\sim\omega}\left[V_p^\pi(x)\right]$.

Define d_p^π as the stationary distribution of the Markov chain that results from following policy π under transition model $p \in \mathcal{P}$. We can show that the above objective $\bar{J}(\pi)$ can be written as an expectation of the reward over a stationary distribution. Indeed, define the average transition model as $\bar{p} := \mathbb{E}_{p\sim\omega}[p]$. Under proper assumptions, it admits a unique stationary law, which will be denoted by $\bar{d}^\pi$ [4].

The goal is now to learn a policy that maximizes the SR average reward $\bar{J}$. We use a policy-gradient method and consider a class of parameterized stochastic policies π_θ with $\theta \in \mathbb{R}^{d_1}$. We will estimate the gradient of the SR objective with respect to θ in order to update the SR policy. The optimal set of parameters thus obtained is denoted by $\theta^* := \arg\max_\theta \bar{J}(\pi_\theta)$. Using the same method as in [17], we can use previous results to derive the gradient of the SR average reward. In order to cope with large state spaces, we can also use a linear approximation $f_w(x,a) := w^T\psi_{xa}$ in place of $\bar{Q}^\pi$ and still point roughly in the direction of the true gradient, as stated below.

Theorem 3.1 (SR Policy-Gradient with Function Approximation)**.** *Let $f_w : \mathcal{X}\times\mathcal{A} \to \mathbb{R}$ be a linear approximator of $\bar{Q}^\pi$. If f_w minimizes the mean squared error $\mathcal{E}^\pi(w) := \sum_{x\in\mathcal{X}} \bar{d}^\pi(x) \sum_{a\in\mathcal{A}} \pi(x,a)[\bar{Q}^\pi(x,a) - f_w(x,a)]^2$ and is compatible in a sense that $\nabla_w f_w(x,a) = \nabla_\theta \log\pi(x,a)$, then $\nabla_\theta \bar{J}(\pi) = \sum_{x\in\mathcal{X}} \bar{d}^\pi(x) \sum_{a\in\mathcal{A}} \nabla_\theta\pi(x,a) f_w(x,a)$.*

4 Soft-Robust Actor-Critic Algorithm

Our SR-AC algorithm is described in Algorithm 1. An uncertainty set $\mathcal{P}$ and a nominal model without uncertainty are provided as inputs. The nominal model is fixed during training. A distribution ω over $\mathcal{P}$ is also provided. The step-size sequences $(\alpha_t, \beta_t, \xi_t; t \geq 0)$ are non-negative numbers properly chosen by the user. At each iteration, samples are generated using the nominal model and the current policy. These are utilized to update the SR average reward (Line 5) and the critic (Line 7) based on an estimate of an SR TD-error. We then exploit the critic to improve our policy by updating policy parameters in the direction of a gradient estimate for the SR objective (Line 8). This process is repeated until convergence which is insured according to the following.

Theorem 4.1. *Under all the previous assumptions, given $\epsilon > 0$, there exists $\delta > 0$ such that for a parameter vector $\theta_t, t \geq 0$ obtained using the algorithm, if $\sup_{\pi_t} \|e^{\pi_t}\| < \delta$, then the SR-AC algorithm converges almost surely to an ϵ-neighborhood of a local maximum of $\bar{J}$.*

5 Numerical Experiments

We demonstrate the performance of soft-robustness on various domains of finite as well as continuous state and action spaces. We used the existing structure of OpenAI Gym environments to run our experiments [3].

Single-step MDP We consider a simplified formulation of the startup vs teaching dilemma described in Section 1. In Figure 1, failing states are denoted by f_i and successful ones are denoted by s_i. The agent is brought back to s_0 once it has reached one of these.

Continuous domains We ran DQN experiments on Cart-Pole, which has a continuous state space and a set of two possible actions. We then ran DDPG algorithms on the inverted pendulum problem, a continuous state domain in which the agent's possible actions belong to a continuous interval $[-a, a]$.

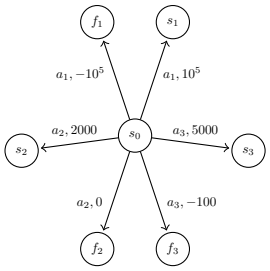


Figure 1: Single-step MDP

5.1 Learning Algorithms

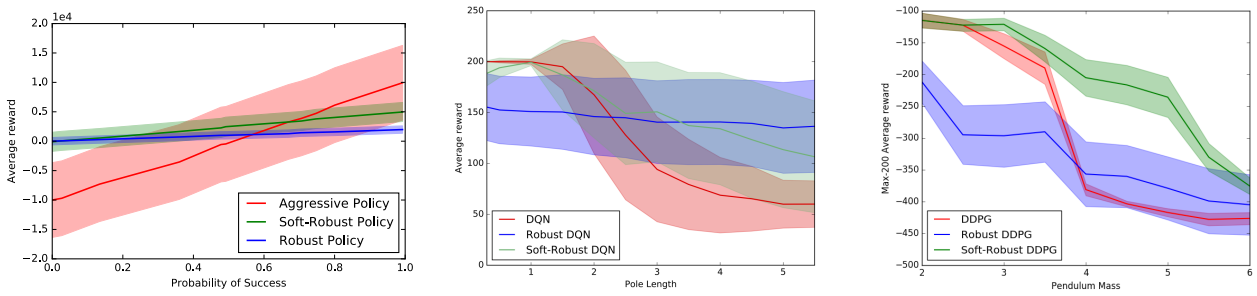
For each experiment, we generate an uncertainty set $\mathcal{P}$ before training. An SR update for the actor is applied by taking the optimal action with respect to the average transition function. We trained the agent on the nominal model in each experiment. In practice, the nominal model is chosen such that a traditional agent attains good performance with a baseline algorithm. The SR agent was learned using SR-AC in the single-step MDP. In Cart-Pole, we run an SR-version of a DQN algorithm. The SR agent in Pendulum was trained using an SR-DDPG.

Figure 2(a) compares the performance of the optimal SR behavior with optimal aggressive and robust policies on the single-step MDP. The three agents are tested against different transition models. As the probability of success gets low, the performance of the aggressive agent drops down below the robust and the SR agents, although it performs best when the probability of success gets close to 1. The robust agent stays stable independently of the parameters but underperforms SR agent which presents the best balance between high reward and low risk.

Algorithm 1 SR-AC

Input: $\mathcal{P}$ - An uncertainty set; $\hat{p} \in \mathcal{P}$ - A nominal model; ω - A distribution over $\mathcal{P}$; f_x - A feature extractor for the SR value function;
Initialize: $\theta = \theta_0$ - An arbitrary policy parameter; $v = v_0$ - An arbitrary set of value function parameters; α_0, β_0, ξ_0 - Initial learning-rates; x_0 - Initial state
repeat
  Act under $a_t \sim \pi_{\theta_t}(x_t, a_t)$
  Observe next state x_{t+1} and reward r_{t+1}
  SR Average Reward Update:
  $\hat{J}_{t+1} = (1 - \xi_t)\hat{J}_t + \xi_t r_{t+1}$
  SR TD-Error:
  $\delta_t = r_{t+1} - \hat{J}_{t+1} + \sum_{x' \in \mathcal{X}} \bar{p}(x_t, a_t, x')\hat{V}_{x'} - \hat{V}_{x_t}$
  Critic Update: $v_{t+1} = v_t + \alpha_t \delta_t \varphi_{x_t}$
  Actor Update: $\theta_{t+1} = \theta_t + \beta_t \delta_t \psi_{x_t a_t}$
until convergence
Return: SR policy parameters θ and SR value-function parameters v

Similar results can be seen in Cartpole and Pendulum (Figures 2(b) and 2(c)) when testing the agents against different pole lengths and pendulum masses respectively. The robust strategy solves the task in a sub-optimal fashion, but is less affected by model misspecification due to its conservative strategy. The aggressive non-robust agent is more sensitive to model misspecification compared to the other methods as can be seen by its sudden dip in performance, below even that of the robust agent in Pendulum. The SR solution strikes a nice balance between being less sensitive to model misspecification than the aggressive agent, and producing better performance compared to the robust solution.



(a) (b) (c)

Figure 2: (a) Single-step MDP: Average reward for AC, robust AC and SR-AC (b) Cartpole: Average reward for DQN, robust DQN and SR-DQN (c) Pendulum: Max-200 episodes average performance for DDPG, robust DDPG and SR-DDPG

6 Discussion

We have presented the SR-AC framework that is able to learn policies which keep a balance between aggressive and robust behaviors. SR-AC requires a stationary distribution under the average transition model which ensures convergence. This is the first work that has attempted to incorporate a soft form of robustness into an online actor-critic method. Our approach has shown promising capability of its scalability to large domains because of its low computational price. The chosen weighting over the uncertainty set can be thought as the way the adversary distributes over different transition laws. In our current setting, this adversarial distribution stays constant without accounting for the rewards obtained by the agent. Future work should address the problem of learning the sequential game induced by an evolving adversarial distribution to derive an optimal SR policy. Other extensions of our work may also consider non-linear objective functions such as higher order moments with respect to the adversarial distribution.

References

[1] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834 – 846, 1983.

[2] Shalabh Bhatnagar, Richard Sutton, Mohammad Ghavamzadeh, and Mark Lee. *Natural Actor-Critic Algorithms*. Automatica, elsevier edition, 2009.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. arXiv:1606.01540v1, 2016.

[4] Esther Derman, Daniel J. Mankowitz, Timothy A. Mann, and Shie Mannor. Soft-robust actor-critic policy-gradient. *AUAI press for Association for Uncertainty in Artificial Intelligence*, pages 208–218, 2018.

[5] Ivo Grondman, Lucian Busoniu, Gabriel A.D. Lopes, and Robert Babuska. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 42(1291-1307), 2012.

[6] Garud N. Iyengar. Robust Dynamic Programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

[7] Vijay R. Konda and John N. Tsitsiklis. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

[8] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous Control with Deep Reinforcement Learning. *arXiv:1509.02971*, US Patent App. 15/217,758, 2016.

[9] Shie Mannor, Ofir Mebel, and Huan Xu. Lightning Does Not Strike Twice: Robust MDPs with Coupled Uncertainty. In *ICML*, 2012.

[10] Shie Mannor, Ofir Mebel, and Huan Xu. Robust MDPs with k-Rectangular Uncertainty. *Mathematics of Operations Research*, 41(4):1484–1509, 2016.

[11] Shie Mannor, Duncan Simester, Peng Sun, and John N. Tsitsiklis. Bias and Variance Approximation in Value Function Estimates. *Management Science*, 53(2):308–322, 2007.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning: Technical Report. *DeepMind Technologies*, arXiv:1312.5602, 2013.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[14] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):783–798, 2005.

[15] Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under Model Mismatch. *31st Conference on Neural Information Processing Systems*, 2017.

[16] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. *ICML*, 2014.

[17] Richard S. Sutton, David McAllester, Satinger Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, volume 12, pages 1057–1063, 2000.

[18] Aviv Tamar, Shie Mannor, and Huan Xu. Scaling up robust mdps using function approximation. *ICML*, 32:1401–1415, 2014.

[19] Huan Xu and Shie Mannor. Distributionally Robust Markov Decision Processes. *Mathematics of Operations Research*, 37(2):288–300, 2012.

[20] Pengqian Yu and Huan Xu. Distributionally Robust Counterpart in Markov Decision Processes. *IEEE Transactions on Automatic Control*, 61(9):2538 – 2543, 2016.

Modeling the development of learning strategies in a volatile environment

Maria K Eckstein
Department of Psychology
UC Berkeley
Berkeley, CA 94720
maria.eckstein@berkeley.edu

Ron Dahl
Department of Public Health
UC Berkeley
Berkeley, CA 94720
rondahl@berkeley.edu

Linda Wilbrecht
Department of Psychology
UC Berkeley
Berkeley, CA 94720
wilbrecht@berkeley.edu

Anne GE Collins
Department of Psychology
UC Berkeley
Berkeley, CA 94720
annecollins@berkeley.edu

Abstract

The development of cognitive abilities is tightly linked to developmental changes in the underlying neural substrate. The current research assesses the relationship between learning and decision making in a volatile environment, and age-related developments in cognition and brain maturation. 322 participants aged 7-18 and 25-30 were tested in several learning and decision making tasks, one of which is the focus of this paper. This probabilistic switching task required participants to select a correct action based on probabilistic feedback, whereby the correct action changed unpredictably. We found that, out of all age groups, children aged 7-12 switched their behavior most rapidly, which was reflected in good short-term, but sub-optimal long-term performance. Adolescents aged 13-18, on the other hand, showed the most persistent choices of all age groups, reflected in suboptimal short-term but close-to-optimal long-term performance. Young adults (25-30) showed intermediate behavior. We employed a reinforcement learning model to assess the underlying mechanisms and found that the inverse-U shaped behavioral changes were captured by a model in which multiple individual parameters changed linearly with age. Specifically, decision noise decreased continuously into adulthood and choice persistence increased continuously. The learning rate from negative feedback, on the other hand, had stabilized by adolescence. These findings are in accordance with the sequential development of cortical and sub-cortical brain regions. Future analyses will assess the role of pubertal hormones in behavioral strategies and computational models.

Keywords: development, reinforcement learning, decision making, cognitive control, computational modeling, puberty, testosterone, brain development

Acknowledgements

Funding for the research was provided by NSF fellowhip SL:CN 1640885.

1 Introduction

Learning and adjusting behaviors to new circumstances is crucial for successful behavior in volatile environments. The current study assesses how this ability develops and relates to brain maturation. Changes in cognitive processing have been associated with brain myelination (increases in white matter), starting before birth and progressing linearly until around age 20 (Giedd et al., 1999). Brain development is also characterized by the formation and subsequent loss of synapses and neurons (changes in grey matter), whereby different brain regions undergo this inverted-U process at different rates. Grey matter maturation progresses from lower-level cortical areas (e.g., sensory and motor cortex, age 4-8) to higher-level ones (e.g., association cortex, young adulthood; Gogtay et al., 2004; Sowell et al., 2003). Subcortical regions (e.g., basal ganglia) are still developing during late adolescence (Thompson et al., 2000; Dennison et al., 2013).

The maturation of higher-level cortical regions has been associated with increases in IQ (Sowell et al., 2003), and we hypothesized that the maturation of subcortical regions–implicated in reinforcement learning and decision making (Niv, 2009)–was similarly linked to the development of specific cognitive capacities. To assess this hypothesis, we tested participants aged 8-18 and 25-30 on several learning tasks spanning the dimensions of volatile / stable environments and stochastic / deterministic feedback. Here, we focus on the volatile probabilistic task, in which participants learned through trial and error which one of two stimuli was rewarding at any given moment. Once participants selected the rewarding stimulus persistently, it changed, forcing participants to switch behavior. Probabilistic feedback precluded absolute certainty as to which stimulus was the correct one.

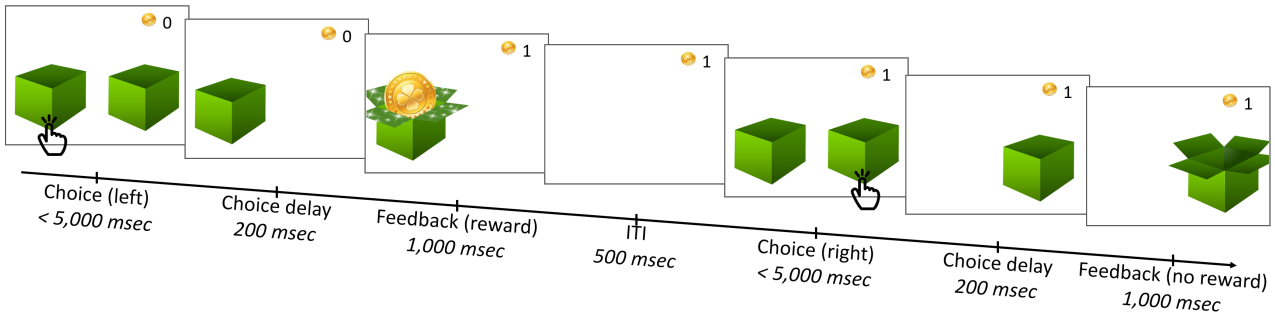


Figure 1: Task design. Participants saw two boxes and were given up to 5 sec to choose one using a game controller. The unchosen box disappeared upon selection, then the chosen box either opened to reveal a golden coin (reward), increasing a coin counter, or was empty (no reward). Feedback was shown for 1 sec, then a 0.5 sec inter-trial interval (ITI) followed.

2 Methods

Participants We tested a sample of 233 participants in this study: 93 children (ages 8-12), 86 adolescents (ages 13-18), and 54 adults (ages 25-30). All participants were recruited from the community, using protocols approved by the institutional review board of the University of California, Berkeley. All participants were free of present or past psychological and neurological disorders. Compensation consisted in 50$ for the 2-hour in-lab portion of the study, and additional 25$ for the completion of take-home saliva samples.

Experimental details All participants completed four computerized experimental tasks, three psychological questionnaires, and a saliva sample during their lab visit, which lasted from about 60 minutes for adults to 120 minutes for children. The probabilistic switching task was the 4th task and consisted of 150 trials as exemplified in Fig. 1. On each trial, participants selected between two boxes. One of the boxes was correct at any point in time (75% probability of reward), whereas and the other one was incorrect (0% probability of reward). After participants had earned 7-15 rewards for the same box, contingencies switched without notice, such that the previously incorrect box was suddenly correct. Switches only occurred after rewarded trials and the first correct trial after a switch was always rewarded. Participants underwent 2-9 switches during the 200 trials ($mean = 7.26, sd = 1.02$).

Behavioral analysis We specified regression models using the R package lme4. Mixed-effects models were created with random-effects of subjects and gender ($+(1|Subject) + (1|Gender)$), or blocks ($+(1|block)$), depending on the model. Reaction times (RTs), and testosterone levels were log-transformed to render their distribution normal. Predictor variables were z-scored to facilitate model fitting.

RL model We modeled behavior in this task using a reinforcement learning algorithm (RL; Sutton and Barto, 2017). Agents learned the value of choosing the left or right box. The value of $a \in a_{left}, a_{right}$ was updated based on the

standard delta rule, $Q(a) = Q(a) + \alpha(r - Q(a))$. Learning rates for positive ($r = 1$) and negative outcomes ($r = 0$) were modeled independently: $\alpha \in \alpha_{pos}, \alpha_{neg}$. Furthermore, the model allowed for counter-factual updating of non-selected actions a_{ns}, as only one action was correct at any moment: $Q(a_{ns}) = Q(a_{ns}) + c\,\alpha(1 - r - Q_{ns})$, where $0 < c < 1$.

Action probabilities $p(a)$ were calculated using a softmax transform on action values with free parameter β, the decision temperature or inverse noise. Importantly, instead of modeling choices between the left and right box, we modelled choices between staying (repeating the previous trial's choice) and switching (picking the other box). The value of staying/switching was given by the value of the corresponding box. This allowed us to define the parameter d, the undecision point. Action probabilities were calculated using $p(a) = \frac{1}{exp(-\beta(d-0.5+Q(other)-Q(a))}$. Thus, d indicates the minimum value $Q(a_{stay})$ at which a_{stay} was selected with higher probability than a_{switch}. A value of $d = 0.5$ led to selecting a random action in the case of equal-valued choices. $d < 0.5$, on the other hand, led to a preference for staying, and $d > 0.5$ led to a preference for switching, in the case of equal values. We assessed simpler models with fewer parameters, and used model comparison to identify the best model. The winning RL model had five free parameters: α_{pos}, α_{neg}, c, β, and d.

Model fitting We fit the RL model in a Bayesian framework, i.e., estimating the full posterior probability distributions for all parameters θ given the human data d: $p(\theta|d) \propto p(d|\theta)p(\theta)$. The Bayesian framework provided two main advantages compared to, e.g., maximum likelihood fitting. First, posterior distributions included information about uncertainty in parameter estimation. Second, the Bayesian approach facilitated hierarchical model fitting, allowing us to test parameter differences between age groups within the model.

We employed Markov Chain Monte Carlo (MCMC) to approximate posterior distributions, using the no-U-turn (NUTS) sampler provided by the python package PyMC3. We identified the winning model described above by fitting several candidate models that differed in terms of included parameters, and selected the one with the best WAIC score. The Bayesian model had the following structure. We modeled separate distributions for each age group (children, adolescents, adults), from which individuals' parameters were drawn, reflecting the notion that parameters might differ between age groups. Parameters for the age groups were drawn from a common distribution, reflecting the fact that age groups shared common priors. Individual parameters α_{pos}, α_{neg}, and c were sampled from Beta distributions; β was sampled from Gamma distributions, and d was sampled from a Normal distribution. Age group distribution parameters were drawn from Gamma distributions, whose parameters were sampled uniformly.

3 Results

3.1 Behavioral analyses

We first assessed the relationship between age and task performance in participants aged 7-18. A logistic regression model predicting task accuracy (selecting the correct box independent of obtained feedback) from z-scored age revealed a significant increase in accuracy with age, $\beta = 0.12, z = 4.84, p < 0.001$. Similarly, linear regression on log-transformed RTs (correct trials only) revealed speeding of RTs with age, $\beta = -0.18, t(183) = -8.53, p < 0.001$.

We next aimed to shed light on the cognitive processes that underlay these improvements in task performance. We first assessed switch trials, i.e., trials in which the previously rewarded box became the non-rewarded box. We found that children aged 7-12 were more likely to change their behavior on the first trial after a switch than adolescents aged 13-18, $t(183) = 4.0, p < 0.001$ (Fig. 2A). Rapid switching is the optimal behavior on switch trials, but rapid switching also leads to mistakes on the long run because many unrewarded trials are not switch trials. Indeed, children showed worse long-term performance (trials 3-7 after a switch) than adolescents, $t(181) = -5.5, p < 0.001$. Strikingly, adults showed intermediate performance between children and adolescents for both short-term (trial 1) and long-term performance (trials 3-7), highlighting the striking differences between childhood and adolescence (trial 1, adults compared to children: $t(113) = 2.22, p = 0.028$; to adolescents: $t(109) = -1.17, p = 0.25$; trials 3-7, adults compared to children: $t(108) = -2.1, p = 0.035$; to adolescents: $t(107) = 2.4, p = 0.017$).

We next assessed participants' responses as a function of the two most recent actions and outcomes. When rewards were followed by no reward ("reward, no reward", Fig. 2B), the same inverse-u pattern with age arose as on the first trial after a switch (Fig. 2A). In all other conditions, the percentage of staying increased with age, evident in significant effects of age on staying in linear regression models ("both reward": $\beta = 0.27\%, t(202) = 3.39, p < 0.001$; "no reward, reward": $\beta = 0.51\%, t(253) = 4.29, p < 0.001$, "both no reward": $\beta = 0.19\%, t(258) = 1.34, p = 0.19$).

Lastly, we assessed the short- and long-term effects of positive and negative feedback on behavior. We used logistic regression to predict participants' actions on trial t from actions and outcomes on trial $t-i$, with separate models for each $1 < i < 9$ and each participant. The models predicted actions (right: 1; left: -1) from "reward" (no reward: 0; reward-right: 1; reward-left: -1) and "noReward" (reward: 0; no-reward-right: 1; no-reward-left: -1). The models revealed large effects of positive feedback for all age groups, showing that being rewarded increased participants' probability of reselecting the same action. This effect slowly diminished over time (Fig. 2C right). T-tests revealed age differences in

that positive feedback had smaller effects on children than adolescents on trials $t-1$ to $t-5$, all $ts < -2.14, ps < 0.033$. Negative feedback affected action selection in that participants were less likely to reselect actions that were not rewarded, although these effects were much smaller than for positive feedback (Fig. 2C left). An interesting pattern emerged for 1-back negative feedback. Children exhibited a large negative effect, supporting the notion of fast switching in the face of negative feedback. Adolescents' effect of 1-back negative feedback, on the other hand, did not differ from 0, $t(85) = -1.33, p = 0.19$, suggesting that adolescents were able to ignore single negative outcomes in the strategic pursuit of rewards. The differences between children and adolescents was significant, $t(148) = -3.37, p < 0.001$, and adults showed intermediate behavior.

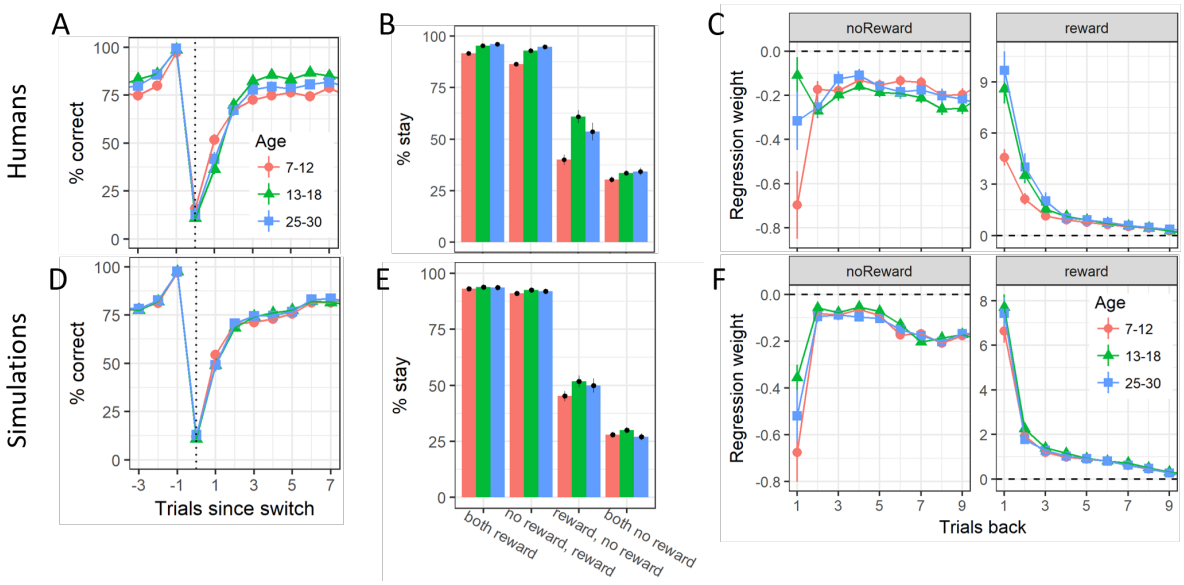


Figure 2: Task performance. (A) Accuracy following switch trials. Children aged 7-12 switched fastest but reached the worst long-term performance, whereas adolescents aged 13-18 switched slowest but achieved the best long-term performance. (B) Effect of the two previous outcomes (x-axis) on staying behavior (i.e., repeating the same action). In general, staying increased with age. The "reward, no reward" condition replicated part A. (C) Results of a logistic regression model predicting choice from several previous actions and outcomes (see main text). Negative feedback ("noReward") had small effects compared to positive feedback ("reward"). (D)-(E) Same analyses for the best-fit RL model.

3.2 RL model

We next turned to our computational model to investigate potential mechanisms behind these differences. Simulations based on the best-fitting RL model showed qualitatively similar–albeit quantitatively smaller–effects than in human participants (see discussion; Fig. 2D-F). The model revealed that age groups did not differ in terms of learning rate from positive feedback α_{pos} or the parameter c for counter-factual learning (table 1). The softmax parameters β and d, on the other hand, changed with age, such that decision noise decreased and undecision points shifted toward staying, in accordance with the behavioral results (table 1). Children were more sensitive to negative rewards (α_{neg}) than both adolescents and adults, as suggested by the behavioral analyses.

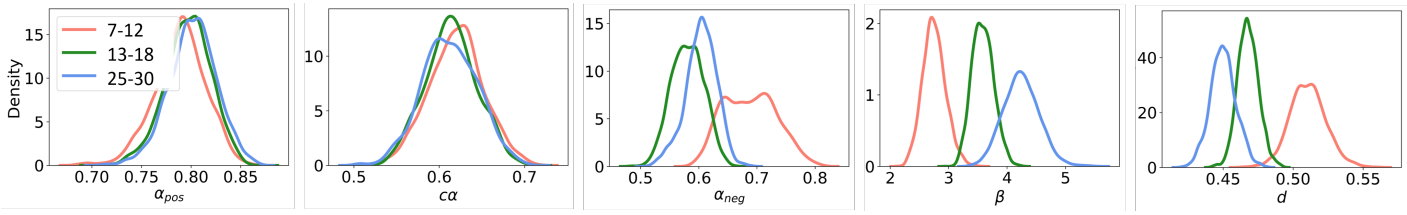


Figure 3: Modeling results. Parameters of the RL model were fit in a hierarchical Bayesian framework. The figure shows the posterior probabilities over parameter means for each age group.

Table 1: Parameter differences based on age groups. p-values denote the percentage of MCMC samples in which the parameter mean of one group was larger than in another. p-values $p < 0.05$ highlighted with stars.

p-values	**Children vs adolescents**	**Children vs adults**	**Adolescents vs adults**
α_{pos}	0.35	0.27	0.43
c	0.63	0.62	0.52
α_{neg}	0.0046 *	0.029 *	0.80
β	< 0.001 *	< 0.001 *	0.0188 *
d	0.0042 *	< 0.001 *	0.067 ·

4 Discussion

We presented a computational model that captures non-linear behavioral differences between children, adolescents, and adults in a probabilistic switching task, with the goal of shedding light on cognitive changes associated with age-related brain maturation. Our RL model captured key aspects of human behavior, such as an inverse-U shape of 1-trial switching, smaller effects of positive feedback in children than adolescents and adults, and larger effects of negative feedback in children than adults, which was in turn larger than in adolescents. Albeit qualitatively similar, these effects were quantitatively smaller than in humans and we will, in future work, explore a larger number of alternative models, such as Bayesian inferences and the combination of RL with task-specific strategic components, to specifically capture adolescents' behavior better.

Children's behavior was marked by rapid switches in response to negative feedback and worse long-term performance, and was reflected in reduced learning rate from negative feedback and increased decision noise in the computational model. Adolescents, on the other hand, showed a marked ability to overrule negative feedback for a single trial and showed close-to-optimal long-term performance, reflected in smaller learning rates from negative feedback and a general shift toward staying in the undecision point parameter. Interestingly, adult behavior—intermediate between children and adolescents—was not associated with intermediate model parameters. Instead, parameters changed linearly with age. Thereby, learning rate from negative feedback decreased from childhood to adolescence, but was constant thereafter, consistent with an early maturation of basal ganglia regions. Decision noise and non-decision point, on the other hand, continued changing into adulthood, consistent with the continued maturation of higher-level cortical regions.

Taken together, our results suggest that non-linear developmental changes can arise from linear changes in the underlying computational and neural processes, highlighting the benefits of computational modeling for developmental cognitive neuroscience. Our future work will explore these results in more depth, to explore potential effects of pubertal onset and / or pubertal status. Neural development can be sensitive to hormonal changes and pubertal hormones and markers have been linked to structural and functional reorganization in the brain (Blakemore, Burnett, and Dahl, 2010). We will therefore investigate the relationship between pubertal hormones, task performance, and model parameters, hoping to shed light on its underlying mechanisms.

References

Blakemore, S.-J., Burnett, S., & Dahl, R. E. (2010). The Role of Puberty in the Developing Adolescent Brain. *Human Brain Mapping*, *31*(6), 926–933. doi:10.1002/hbm.21052

Dennison, M., Whittle, S., Yucel, M., Vijayakumar, N., Kline, A., Simmons, J., & Allen, N. B. (2013). Mapping subcortical brain maturation during adolescence: Evidence of hemisphere- and sex-specific longitudinal changes. *Developmental Science*, *16*(5), 772–791. doi:10.1111/desc.12057

Giedd, J. N., Blumenthal, J., Jeffries, N. O., Castellanos, F. X., Liu, H., Zijdenbos, A., … Rapoport, J. L. (1999). Brain development during childhood and adolescence: A longitudinal MRI study. *Nature Neuroscience*, *2*(10), 861–863. doi:10.1038/13158

Gogtay, N., Giedd, J. N., Lusk, L., Hayashi, K. M., Greenstein, D., Vaituzis, A. C., … Thompson, P. M. (2004). Dynamic mapping of human cortical development during childhood through early adulthood. *Proceedings of the National Academy of Sciences*, *101*(21), 8174–8179. doi:10.1073/pnas.0402680101

Niv, Y. (2009). Reinforcement learning in the brain. *Journal of Mathematical Psychology*, *53*(3), 139–154.

Sowell, E. R., Peterson, B. S., Thompson, P. M., Welcome, S. E., Henkenius, A. L., & Toga, A. W. (2003). Mapping cortical change across the human life span. *Nature Neuroscience*, *6*(3), 309–315. doi:10.1038/nn1008

Sutton, R. S. & Barto, A. G. (2017). *Reinforcement Learning: An Introduction* (2nd ed.). Cambridge, MA; London, England: MIT Press.

Thompson, P. M., Giedd, J. N., Woods, R. P., MacDonald, D., Evans, A. C., & Toga, A. W. (2000). Growth patterns in the developing brain detected by using continuum mechanical tensor maps. *Nature*, *404*(6774), 190–193. doi:10.1038/35004593

Efficient Count-Based Exploration Methods for Model-Based Reinforcement Learning

Nicolas El Maalouly
School of Engineering
École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland
nicolas.elmaalouly@epfl.ch

Wulfram Gerstner
School of Computer and Communication Sciences and
Brain Mind Institute, School of Life Sciences
École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland
wulfram.gerstner@epfl.ch

Johanni Brea
School of Computer and Communication Sciences and
Brain Mind Institute, School of Life Sciences
École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland
johanni.brea@epfl.ch

Abstract

A key technique to efficient exploration in reinforcement learning is the propagation of reward exploration bonus throughout the state and action space. Adding reward bonuses however, makes the MDP non stationary and requires resolving the MDP after every iteration which can be computationally intensive. Prioritized sweeping with small backups can greatly reduce the computational complexity required for keeping the model and value functions up to date in an online manner. We first propose to adapt exploration bonus techniques to the small backups algorithm in order to achieve better computational efficiency while retaining the benefits of a model-based approach. We then argue for the advantages of maintaining separate value functions for exploration and exploitation and propose different ways of using the two, and also discuss the different properties we get by choosing different forms for the bonus beyond the popular $\frac{1}{\sqrt{n}}$. Finally we present a more general PAC-MDP sample complexity analysis for count-based exploration bonuses. The result is a generalization of count-based exploration methods that can be combined with state tabulation to augment any deep reinforcement learning method with a theoretically justified and efficient model-based approach to exploration.

Keywords: exploration bonus, reinforcement learning, MDP, prioritized sweeping, PAC-MDP, sample complexity, model-based RL

Acknowledgements

This work was supported by the Swiss National Science Foundation (Grant 200020_165538 /"Synaptic Plasticity in System Models").

1 Introduction

The field of reinforcement learning has enjoyed great successes in the past few years, especially when combined with deep learning techniques [1]. Exploration efficiency remains however a big challenge. Model-based techniques like Model-Based Interval Estimation with Exploration Bonus (MBIE-EB) [2] are simple to implement, allow for deep exploration [3] by efficient propagation of the count bonuses, and have theoretical sample complexity guarantees, but their large computational complexity prohibits many interesting applications. Exploration bonus techniques have been recently adapted to large scale problems with neural network function approximation [4, 5], but their use is mostly limited to model-free algorithms which tend to have a higher sample complexity because they fail to propagate the reward bonus fast enough to earlier states.

While targeting lower sample complexity with model-free techniques would be a good way to go, we choose to focus on the other side of the problem: that of achieving better computational efficiency using model-based algorithms. One particularly efficient model-based technique we focus on in this paper is Prioritized Sweeping with Small Backups[6].

2 Background

2.1 Reinforcement Learning

In this paper we use the general formalism of reinforcement learning based on Markov Decision Processes (MDPs), which can be described as a tuple $\langle S, A, P, R, \gamma \rangle$ consisting of S, the set of all states; A, the set of all actions; $P_{sa}^{s'} = Pr(s'|s,a)$, the transition probability from state $s \in S$ to state s' when action $a \in A$ is taken; $R_{sa} = E[r|s,a]$, the expected reward function when action a is taken in state s; and γ, the discount factor. The goal is to find a policy $\pi^\star$ that maximizes the total accumulated reward. An action value function that quantifies the quality of the policy can be defined as $Q^\pi(s,a) = \mathbb{E}\{\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s, a_0 = a, \pi\}$ and a state value function as $V^\pi(s) = Q^\pi(s, \pi(s))$. In a model-based setting, the algorithm keeps track of an estimate of R_{sa} and $P_{sa}^{s'}$ and finds the optimal value function (which can lead to an optimal policy of the form $\pi(s) = \arg\max_a Q(s,a)$) using techniques such as value iteration.

2.2 Prioritized Sweeping with Small Backups (PSSB)

Prioritized sweeping [7] is an efficient way to approximate the value iteration algorithm by limiting the number of backups performed per time step, and prioritizing the ones that are expected to cause large value changes. In [6] the authors greatly improve the efficiency of prioritized sweeping by introducing a new backup method, the "small backup", which uses only the current value of a single successor state and has a computation time independent of the number of successor states. For such a backup to be equivalent to the normal full backup the following relation has to hold:

$$Q(s,a) = R_{sa} + \gamma \sum_{s'} P_{sa}^{s'} V(s') \tag{1}$$

This requires that after each observation of a sample (s, r, s') the following update needs to be performed:

$$N_{sa} \leftarrow N_{sa} + 1; \quad N_{sa}^{s'} \leftarrow N_{sa}^{s'} + 1$$
$$Q(s,a) \leftarrow [Q(s,a)(N_{sa} - 1) + R_{sa} + \gamma V(s')]/N_{sa} \tag{2}$$

Equation 2 implies that the Q-function computed by the algorithm is a running average of the reward plus expected next state value (note that when $V(s')$ is updated later, the Q-function is also updated in the background by the small backups mentioned previously). By using the fact that the true reward can be obtained as the limit of the average observed reward, the PSSB algorithm is able to maintain the best estimate for the value function at every time step through small incremental updates that can be efficiently performed.

2.3 Count-Based Exploration Bonus

Exploration is a fundamental part of reinforcement learning, the agent needs to find a good balance between taking the actions that are so far most promising in terms of maximizing reward, and actions that it has a lot of uncertainty about their outcome since these might still end up being more rewarding. To this end, most reinforcement learning algorithms use action randomization techniques (e.g. ϵ-greedy) to give a non-zero probability to trying out new actions and achieve good exploration in the limit. These techniques, however, may lead to inefficient behavior by sometimes taking actions that the agent is certain are not optimal, and this is known as the problem of undirected exploration. Directed exploration techniques on the other hand, favor actions that may still turn out to be optimal, such as actions that are not yet well explored. Count-based exploration bonus methods achieve directed exploration by favoring less visited state-action pairs through the addition of a reward bonus of the form $\frac{1}{(N_{sa})^\theta}$, where N_{sa} is the number of times the agent took action

a in state s, for all the visited state action pairs, and $\theta > 0$ is a parameter that indicates how fast the bonus goes to 0. This bonus can then be added to the regular reward for a total reward of $R_{s,a} + \frac{\beta}{(N_{sa})^\theta}$ where $\beta \geq 0$ is a factor that determines that scale of the added bonus.

One way to measure the exploration efficiency is by measuring sample complexity. The sample complexity of an algorithm $\mathcal{A}$ is the number of timesteps t such that the policy at time t is not ϵ-optimal from the current state (formally, $V^{\mathcal{A}_t}(s_t) \geq V^\star_M(s_t) - \epsilon$). We say that $\mathcal{A}$ is PAC-MDP (Probably Approximately Correct in Markov Decision Processes) if, for any ϵ and δ, the per-step computational complexity and the sample complexity of $\mathcal{A}$ are less than some polynomial in the relevant quantities $(|S|, |A|, 1/\epsilon, 1/\delta, 1/(1-\gamma))$, with probability at least 1 - δ. Model Based Interval Estimation - Exploration Bonus (MBIE-EB) [2] is one such PAC-MDP algorithm which uses a particular form of count-based exploration bonus.

3 Prioritized Sweeping with Small Backups - Exploration Bonus

In this section we show how we can adapt the PSSB algorithm to efficiently compute an estimate of the propagated reward bonus.

3.1 Computing the Exploration Bonus

The goal is to compute a reward bonus in the form of $\frac{1}{(N_{sa})^\theta}$. PSSB however, keeps track of a running average of the reward incurred in each state-action pair, so simply adding the reward bonus to the regular reward would not work because the resulting bonus would be an average of the received bonuses which is not what we want. In order to use the same equations for computing the reward bonus we need to define a new exploration reward function which results in the desired exploration bonus. We use the following reward function:

$$R^e_{sa} = 1 \text{ if } N_{sa} = 0 \text{ and } \frac{1}{(N_{sa}+1)^{\theta-1}} - \frac{1}{(N_{sa})^{\theta-1}} \text{ otherwise,}$$

which results in the average reward: $\overline{R^e_{sa}} = \frac{1}{(N_{sa})^\theta}$. To implement it with PSSB, we simply need to maintain a separate value function for exploration initialized to:

$$Q_e(s,a) = \frac{1}{1-\gamma_e} = 1 + \gamma_e \frac{1}{1-\gamma_e} \,,$$

which verifies equation 1, and update it in the same way we update the normal value function using equation 2 and a separate priority queue. In the next section we will discuss how to make use of this new value function for exploration.

4 Benefits of Maintaining Separate Value Functions for Exploration and Exploitation

Having a separate value function for exploration can have some advantages over adding the reward bonus directly to the extrinsic reward as it enables different learning schemes for the two value functions (e.g. different discount factor, episodic vs non-episodic updates...) and also allows for different ways of using the exploration value function apart from the general case of linearly combining it with the normal value function (the latter resulting in a behavior equivalent to adding the reward bonus directly to the extrinsic reward). The approaches discussed here can be more generally applied to any algorithm that maintains a separate value function.

4.1 Pure exploration for training, and pure exploitation for testing

Here the agent follows a policy maximizing the exploration value function during training (while still updating the regular value function separately), and then at test time, it uses a policy that maximizes the regular value function. In this setting the advantage of having separate value functions for exploration and exploitation becomes most apparent as it allows the agent to use its best estimate of the value function at any time without needing to wait for bonuses to vanish. Note that using a different policy for training and testing only works in the model-based setting, and would suffer from off-policy issues in a model-free setting.

4.2 Using a policy that maximizes a mixture of value functions

In this case the agent follows a policy of the form:

$$\pi(s) = \arg\max_a (Q(s,a) + \beta(Q_e(s,a))^\alpha)$$

with $\alpha > 0$ and $\beta \geq 0$ mixture parameters, and using on-policy updates for the value functions. Some values of α and θ result in a behavior equivalent to existing algorithms in the literature. For example $\theta = \frac{1}{2}$ and $\alpha = 1$ yields propagation of $\frac{1}{\sqrt{n}}$, which is equivalent to the MBIE-EB algorithm [2]. However, in [8], the authors argue that propagating $\frac{1}{\sqrt{n}}$ is equivalent to summing up standard deviations along a trajectory which is not desirable from a probabilistic stand point, and instead the variance should be propagated which can be transformed into a standard deviation by taking the square root at the end. With $\theta = 1$ and $\alpha = \frac{1}{2}$ we get what the authors propose for the tabular case: propagating $\frac{1}{n}$ which is proportional to the local variance of the value function in the Bayesian setting with Gaussian prior on rewards and Dirichlet prior on the transition function. MBIE-EB, however, still has the advantage of being a PAC MDP (probably approximately correct in MDP) algorithm, while other parameter settings lead to algorithms that lack the theoretical backup. For this reason, in section 5 we extended the sample complexity proof of MBIE-EB to work with other settings of α and θ.

4.3 Determinization of Stochastic Policies

In [9], the authors propose to use counts to approximate stochastic decision making rules (e.g. ϵ-greedy, softmax...) with deterministic policies. The idea is to take

$$\pi(s) = \arg\max_a \left(\frac{f(a|s)}{C(s,a)} \right) \quad \text{e.g. } f(a|s) = \text{softmax}(Q(s,a))$$

where $\frac{1}{C(s,a)}$ is a generalized notion of counts. In their paper they use E-values which generalize counts in a model-free setting. In our case we can use $\frac{1}{C(s,a)} = Q_e(s,a)$ which is a model-based version of generalized counts.

5 Theoretical Analysis

In this section we show that the same sample complexity bound proven by [2] applies to all values of $\theta > 0$ and $0 < \alpha \leq 1$ for a particular choice of β. We also assume that $0 \leq R \leq 1$[1]. Due to lack of space, we omit the complete proof, but briefly discuss the main points and the conditions for it to apply. The main idea is to use theorem 10 from [10] which requires that the value function used to derive the policy is both optimistic and bounded. We get the former by choosing:

$$\beta' = \left(\frac{\beta^{2\theta}}{\alpha \epsilon^{2\theta-1} (1-\gamma)^{2\theta-\alpha}} \right)^{\alpha} \text{ if } \theta > 1/2 \text{ otherwise } \beta' = \beta$$

$$\text{with } \beta = \frac{1}{1-\gamma} \sqrt{\ln(2m|S||A|\delta)/2} \text{ being the constant used for MBIE-EB}$$

and the latter by using

$$\pi(s) = \arg\max_a Q(s,a) \text{ with } Q(s,a) = \min\left(Q_r(s,a) + \beta' \left(Q_e(s,a)\right)^{\alpha}, Q_{max}\right) \text{ and } Q_{max} = \frac{1}{1-\gamma}$$

Figure 1 provides some intuition for how β' is chosen to guarantee optimism for the simple case of $\alpha = 1$ [2]. Since $\frac{\beta}{\sqrt{n}}$ was shown in [2] to guarantee optimism, any bonus bigger than that would also have the same guarantee.

The last condition the algorithm needs to fulfill is that of accuracy, i.e. accurately approximating the true value function. This requires the exploration bonus to be small enough. So after m learning steps, with m being the number of times every state-action pair needs to be visited in order to accurately estimate the value function, we can simply set the reward bonus to 0. We can also show that this does not affect optimism since we only require the value to be within ϵ of the true optimal value which in this case can be ensured by accuracy.

This leads to a PAC-MDP sample complexity of $O\left(\frac{|S|^2|A|}{\epsilon^3(1-\gamma)^6}\right)$, omitting log factors.

It is important to note that in Theorem 2 of [11] the authors proved that using reward bonuses of the form we use in this paper with $\theta > 0.5$ leads to an algorithm that is not PAC-MDP in general. However, their proof relies on taking ϵ as a function of β. In our analysis we take β to be a function of ϵ which still allows for a PAC-MDP algorithm for $\theta > 0.5$, but may result in large values of β. In practice however, values used for β tend to be much smaller than the ones used in the theoretical analysis even in the case of MBIE-EB.

[1] This is a common assumption in the literature and is not very restrictive since any bounded reward can be shifted and re-scaled to verify this condition.

[2] For other values of α however, the analysis gets more complicated since the full exploration bonus added to the regular value function becomes a nonlinear transformation of the count bonuses along the entire trajectory, which means that we cannot treat the count bonuses for every state-action pair separately.

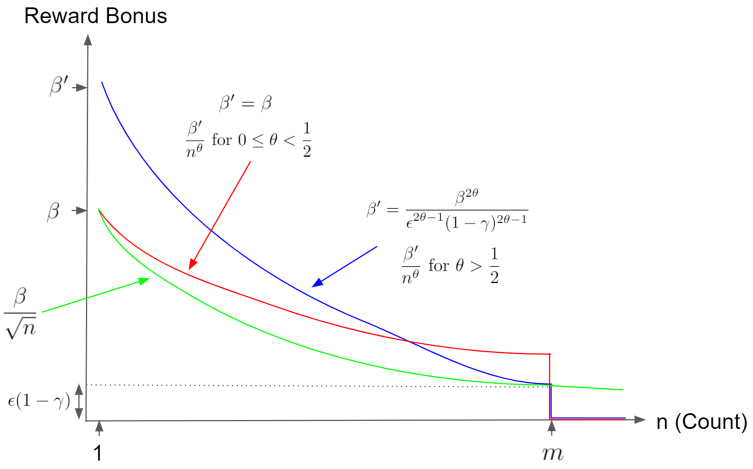


Figure 1: Reward bonus as a function of the visit count for a single state-action pair.

6 Conclusion

This work presented a generalization of count-based exploration bonus techniques that accounts for different forms of bonuses, which can differ in statistical properties, and different ways of balancing exploration and exploitation. We extended the validity of existing sample complexity guaranties to account for a bigger class of these methods, and provided a way for adapting these methods for use with efficient model-based reinforcement learning methods like prioritized sweeping with small backups. While these efficient techniques only work in a tabular setting, recent work shows that they can still be beneficial in high dimensional settings by transforming the environment into a tabular one [4, 12]. Combining such tabulation techniques with our exploration bonus implementation can bring model-based exploration efficiency to high dimensional environments. In contrast to previous work on exploration bonuses with function approximation [4, 5], we propose a model-based approach to exploration that allows to propagate the exploration bonuses more effectively at little additional computational cost.

References

[1] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529.

[2] Strehl, Alexander L., and Michael L. Littman. "An analysis of model-based interval estimation for Markov decision processes." Journal of Computer and System Sciences 74.8 (2008): 1309-1331.

[3] Osband, Ian, et al. "Deep exploration via randomized value functions." arXiv preprint arXiv:1703.07608 (2017).

[4] Tang, Haoran, et al. "# Exploration: A study of count-based exploration for deep reinforcement learning." Advances in Neural Information Processing Systems. 2017.

[5] Bellemare, Marc, et al. "Unifying count-based exploration and intrinsic motivation." Advances in Neural Information Processing Systems. 2016.

[6] Van Seijen, Harm, and Richard S. Sutton. "Efficient planning in MDPs by small backups." Proceedings of the 30th International Conference on International Conference on Machine Learning. Vol. 28. 2013.

[7] Moore, Andrew W., and Christopher G. Atkeson. "Prioritized sweeping: Reinforcement learning with less data and less time." Machine learning 13.1 (1993): 103-130.

[8] O'Donoghue, Brendan, et al. "The Uncertainty Bellman Equation and Exploration." arXiv preprint arXiv:1709.05380 (2017).

[9] Fox, Lior, Leshem Choshen, and Yonatan Loewenstein. "DORA The Explorer: Directed Outreaching Reinforcement Action-Selection." (2018).

[10] Strehl, Alexander L., Lihong Li, and Michael L. Littman. "Reinforcement learning in finite MDPs: PAC analysis." Journal of Machine Learning Research 10.Nov (2009): 2413-2444.

[11] Kolter, J. Zico, and Andrew Y. Ng. "Near-Bayesian exploration in polynomial time." Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009.

[12] Corneil, Dane, Wulfram Gerstner, and Johanni Brea. "Efficient Model-Based Deep Reinforcement Learning with Variational State Tabulation." arXiv preprint arXiv:1802.04325 (2018).

Hidden Information, Teamwork, and Prediction in Trick-Taking Card Games

Hadi Elzayn
School of the Arts and Sciences
University of Pennsylvania
Pennsylvania, PA 19103
hads@sas.upenn.edu

Mikhail Hayhoe
School of Engineering and Applied Sciences
University of Pennsylvania
Pennsylvania, PA 19103
mhayhoe@seas.upenn.edu

Harshat Kumar
School of Engineering and Applied Sciences
University of Pennsylvania
Pennsylvania, PA 19103
harshat@seas.upenn.edu

Mohammad Fereydounian
School of Engineering and Applied Sciences
University of Pennsylvania
Pennsylvania, PA 19103
mferey@seas.upenn.edu

Abstract

We highlight a class of card games which share several interesting features: hidden information, teamwork, and prediction as a crucial component. This family of games in question, known as "Whist" games, consists of games of trick-taking, turn-based play, with team relationships of varying intensities, differing betting and scoring rules, and slight variations in mechanics. Using self-play, we have trained a DeepRL-style algorithm to bet and play *Four Hundred*, a flagship game in the family (*Hearts*, *Spades*, and *Bridge* are all related to varying degrees). Our algorithm reaches human-competitive performance, dominating all baselines it was tested against and learning the importance of key game concepts such as trump and partnership. Moreover, it exhibits reasonable context-specific strategies, suggesting an adaptability of the framework to different scenarios.

We believe this family of games provides an interesting testing ground for reinforcement learning algorithms because of its features; however, we are most interested in developing methods to transfer insights across variations of games. We hope that such an approach will result in more efficient training and perhaps more human-like play.

Keywords: Reinforcement Learning, Self-Play, Games, Transfer Learning

Acknowledgements

We appreciate valuable feedback from Shivani Agarwal, Heejin Chloe Jeong, Simeng Sun, and Steven Chen.

1 Introduction

Inspired by the recent successes in the design of artificial intelligence to play games such as Backgammon [6], Chess [4], Poker [1], Atari games [3], and Go [2] with superhuman performance, we study a family of card games ("Whist" descendants) that provide a rich strategic environment for testing learning algorithms. These games share several interesting features, including hidden information, teamwork, and prediction as crucial components. The Whist family consists of games of trick-taking, turn-based play, with team relationships of varying intensities, differing betting and scoring rules, and slight variations in mechanics. In work so far, we have focused on *Four Hundred*, a flagship of the game family, and design a reinforcement learning algorithm to play the game at a level competitive with humans. We describe our problem formulation, algorithms, and results in Section 2. We are also interested in this family of games as a testing ground to explore the transfer of insights and strategies from one game into similar games. We briefly discuss this agenda in Section 3.

2 Solving *Four Hundred*

Rules of *Four Hundred*

A deck of cards is distributed evenly (face down) to four players. Each player sits across from their teammate. No communication of hands is allowed. Before beginning play, players must bet an expected number of 'tricks' (one round of played cards) they plan to take over the 13-card hand, between 2 and 13. In each round after bets are placed, players take turns choosing a card to play from their hand in order, beginning with the player to take the previous trick. The suit of the first card played determines the 'lead suit', and players must play cards from that suit if possible. The winner of the trick is the player with the highest card of the 'lead suit', or the highest card of the 'trump suit' (Hearts) if any were played. At the end of 13 rounds, each player's score is increased by their bet if they meet or exceed it, but decreased by their bet if they fail to meet it. The game is over once one team has a player with 41 or more points, and the other player has positive points.

Learning Problem *Find an optimal betting policy $\beta^{i,*}$ and playing policy $\pi_t^{i,*}$ to maximize expected reward for player i given each other and the play of the others:*

$$\beta^{i,*} = \arg\max_{\beta \in \mathcal{B}} E\left[\sum_{t=0}^{13} R\left(\beta(\mathcal{H}_0^i), \pi_t^{i,*}(\mathcal{H}_t^i)\right)\right], \qquad \pi_t^{i,*} = \arg\max_{c \in \mathcal{H}_t^i} E\left[R(\beta^{i,*}(\mathcal{H}_0^i), c) \mid S_t^i\right].$$

where $\mathcal{H}_0^i$ is player i's starting hand, $\mathcal{H}_t^i$ is their hand after trick $t \in \{1, \ldots, 13\}$, and S_t^i is the information known to player i about the state of the game up until trick t.

BETTING: Since the final score depends not only on tricks taken, but on the bet made, the betting and playing tasks are deeply intertwined. While both may be viewed from a reinforcement learning perspective, we choose to view betting as a supervised learning problem. Given some particular strategy and initial hand, the player can expect to win some number of tricks (where randomness comes from the distribution of cards across opponents' hands as well as variation in player strategies, stochastic and otherwise). Given observed games under a fixed strategy and some observed initial hand compositions, a model that predicts tricks taken by the end of the round may serve as a good bet model.

Thus, the data we generate during games in the form of initial hands can serve as input data, and the number of tricks won functions as the label. We implement a neural network (NN) for regression using this data. The input is a 4×13 binary matrix with exactly 13 non-zero elements representing which cards are in the player's hand. Each column of the matrix represents a value (e.g. 7 or *Queen*), while each row represents a suit. The first suit is reserved for the trump suit, which is higher in value than all other suits. The output of the neural network is a real number, which we then map to the closest integer between 2 and 13 (since bets of 0 and 1 are not allowed). We define the loss as the squared difference between the score received and the best possible, which is also the squared difference between the optimal and observed reward (see (1)):

$$\ell_{\text{bet}}(y, \hat{y}) = (y - \text{sign}(\hat{y} - y) \cdot \hat{y})^2 = \begin{cases} (y - \hat{y})^2, & \text{if } y \geq \hat{y}, \\ (y + \hat{y})^2, & \text{otherwise,} \end{cases}$$

where $\hat{y}$ is the *bet*, and y is the *tricks won*. Motivated by the game's scoring rules, this loss function is asymmetric - it penalizes more for bets which are higher than the tricks obtained. Therefore our goal in this supervised learning problem is to learn a relationship between the initial cards dealt to each player and the number of tricks that player won at the end of the round. In this regard, the actual play of the game greatly affects the number of tricks expected to win.

CARD PLAY: The second component of the game is the playing of cards, wherein we consider a reinforcement learning approach to design a strategy. We capture the state of the game by three 4×13 matrices and two 1×4 vectors in the following way. The first matrix represents the order history of the game; each card played is represented by an integer between 1 and 52 (initialized to zero), based on the order in which it was played. For example, in the 5th trick of the game, the card played by the third player will have a 23 in the corresponding location (5th trick $\times$ 4 cards per trick + 3rd card played = 23). The second matrix is for player history. As each card is played, its location will be filled by a number denoting the ID of the player. Continuing our example, if the card above was played by player 2, then a 2 will be put in the corresponding location. Representing order and player history as a matrix in this fashion was inspired by the state representation of AlphaGo. The final matrix is the player hand, which is similar to the input for the betting neural network, and indeed is identical at the beginning of the game. As the game continues, whenever a card is played, the 1 indicating the presence of a card in the hand becomes a zero. Returning to our example, the matrix after the fifth round will have 8 (13 initial cards - 5 cards played) non-zero elements. The first 1×4 vector contains the bets that each player made at the beginning of the hand, and the second contains the tricks that each player has won so far.

Given the states, we define a reward at the end of each round by

$$\text{Reward} = \begin{cases} \text{bet}, & \text{if tricks} \geq \text{bet}, \\ -\text{bet}, & \text{otherwise.} \end{cases} \tag{1}$$

The state size is combinatorially large, and hence we do not consider tabular reinforcement learning solution methods but rather function approximation with neural nets. Ultimately, given the possible actions available to the player, we will evaluate the function at each of the potential states and choose the action which enters the state with the highest value. Our approach is informed by the classical Q-learning approach, where the value function is updated by

$$V^{t+1}(s) = \max_a \sum_{s'} p(s'|s,a)(r(s,a,s') + \gamma V^t(s')),$$

where s is the state, a is an action, s' is the state after taking action a from state s, and $\gamma \in [0,1)$ is the discount factor. We consider a mild adaptation, similar to [5], in which the reward is provided as a label to each observed state, and the neural net Q-update occurs in batches. Given that reward is observed at the end of the round, for trick $t \in \{1, \ldots, 13\}$, we assign a reward to that state s by

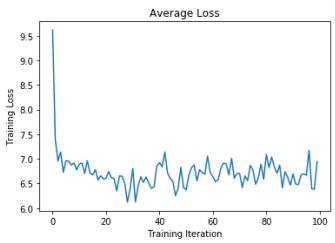
$$\text{Value}(s) = \gamma^{13-t} \cdot (\text{Reward}_{\text{Team member 1}} + \text{Reward}_{\text{Team member 2}}) + \mathbf{1}\{\text{Team won the trick}\}.$$

We include the $\mathbf{1}$ term for reward shaping [3] as it is a favorable outcome which should help increase convergence. Once we have assigned a label to each of the states in terms of the value defined above, we use a neural network for regression to map each state to the appropriate value.

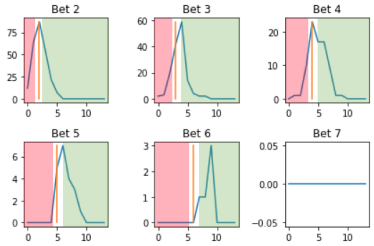
BASELINES: All data was generated from a game simulator developed from scratch. Three baselines were created to compare against: the first baseline is **Random Play-Random Bet**. Random bet selects a random value from $\{2, 3, 4, 5\}$ during the betting stage, and in each round chooses a random card to play from the set of valid cards. The second baseline is **Greedy Play-Model Bet**. During play, greedy simply selects the highest card in its hand from the set of valid cards, without consideration of its partner. Betting for greedy uses a neural net trained on 100,000 games of 4 greedy players with the same architecture as *Over400*, as described earlier. For the final baseline, **Heuristic Play-Model Bet**, a heuristic strategy was defined based on human knowledge of the game. This heuristic player takes into account available knowledge of the team member's actions as well as opponents' actions, and straightforward, if complicated, control flow to determine which move to play. In particular, it keeps track of whether the trump suit has been broken, whether the currently winning player is a teammate or opponent, and what the minimum card guaranteed to win the trick is. The betting for the heuristic baseline is also given by a neural network model which was trained, in the same way as greedy, on 100,000 games of 4 heuristic players.

2.1 Results

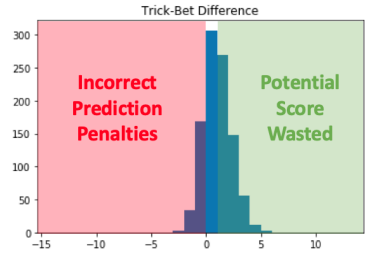
To understand the performance of the betting model, we consider the metrics of loss and Trick-bet difference. The noise observed is due to the betting and playing strategies being constantly updated in tandem. Figure 1c shows the Trick-bet difference metric. As we can see, the highest peaks of the histogram occur at Tricks - Bet $\in \{0, 1\}$. This means that most of the bets are slightly less than the number of actual tricks taken, indicating that the players are betting conservatively.



(a) Loss function for the betting NN, trained in tandem with the playing NN.



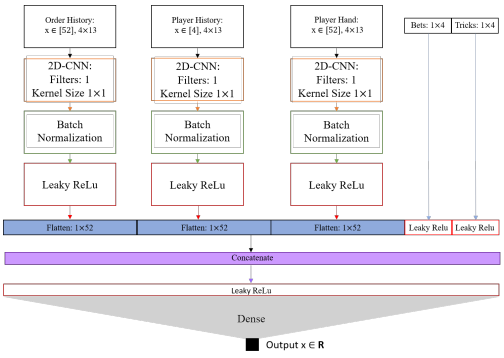
(b) Tricks won over 1,000 games, based on bet made. No bets were higher than 6.



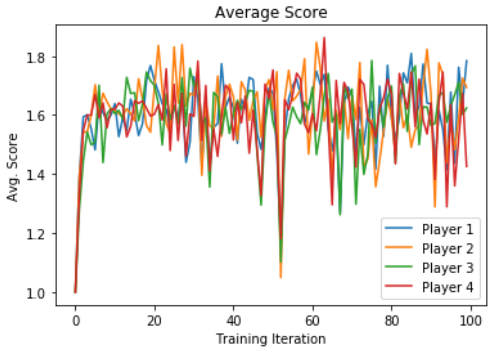
(c) Histogram showing difference between tricks won and bet made.

Figure 1: Betting performance during games of self play. Tricks to the right of the bar (green area) represent wasted potential score, since the AI won more tricks than it bet; tricks to the left of the bar (red area) represent score penalties incurred, since the AI won fewer tricks than it bet. Bets were made conservatively, since the AI made or exceeded its bet over 80% of the time.

To understand the performance of the card play network, we consider the average score shown in Figure 2b. As expected, the average score of all four players increases as the number of training iterations increase, with some noise due to the betting and playing neural networks being trained in tandem. It is important to note that the game is learning how to play relative to the other players, in that it understands the relationship between its team member and its opponents relative to its position. Finally, we consider Table 1 which shows the performance of our Program against the baselines. Our Heuristic algorithm was able to defeat Random and Greedy Baselines 100 - 0, as did *Over400*. Even more impressively, *Over400* **beat the heuristic baseline 89.5 % of the time**.



(a) Neural Network Architecture for Card Play



(b) Average score showing the performance of the self play NN model

Figure 2: Network Architecture and Performance

Human players also squared off against the network to get a sense as to how *Over400* was playing. The AI understood to win tricks by playing high cards of the lead suit and also to play a trump card when the lead suit had run out in its hand. It had also learned to play differently depending on whether or not it had won enough tricks to make its bet (e.g., attempting to lose tricks if it already made its bet). However,

Over400 would occasionally play high cards which were not capable of winning since they were not the lead or trump suit. We imagine this is due to the difficulty in learning the concept of the lead suit, which may have been accomplished with more samples of training data or with a deeper neural net structure.

	Random	Greedy	Heuristic	*Over400*
Heuristic Win %	100	100	-	10.5
Over400 Win %	100	100	89.5	-

Table 1: Percentage of wins against the baselines over 200 games.

3 Future Work

There are over 30 varieties of Whist games with major or minor variations in rules. In future work, we will examine whether and how, by learning one game, we can learn them all. For example, *Spades* is nearly the same as *Four Hundred*: the objective is to take tricks, but the trump suit is Spades (rather than Hearts), and there is usually no betting or teams. *Tarneeb* is nearly the same as *Four Hundred*, but the trump suit is chosen by players (and the betting is more complicated). Finally, *Hearts* has the same mechanics as *Spades*, but the goal is to *avoid* taking tricks.

A policy trained for *Four Hundred* will likely fail to produce good results on any of these other games. However, it is clear that the similarity of structure in these games *should* mean a learner which is good at one game can perform well when playing others. An analogy can be drawn to classification: playing *Spades* well and *Hearts* poorly feels similar to classifying objects with high accuracy but failing to do so if they are rotated and translated. Overcoming this problem in image classification required, in some sense, focusing on features that were 'invariant' to translation and rotation; in our case, there is some 'invariance' about the relative ordering of the cards.

We briefly describe our proposed approach, which we call *invariance discovery*, or *insight abstraction*. Let S be the set of observed states, V_G be the value of each state under game G, and $V_{G'}$ be the value of the state under game G'. Recall that we estimate V_G to be able to return the following policy:

$$\pi_G = \text{argmax}_a \mathbb{E}_{s' \sim P(s,a)} \left[V(s') \right].$$

Instead of retraining for G', if we learn the *invariance mapping* $\chi : V_G(s) \mapsto V_{G'}(s)$, we can then return the policy $\pi_{G'} := \text{argmax}_a \mathbb{E}_{s' \sim P(s,a)} \chi(V(s'))$. Such a mapping exists whenever the value functions do, but learning this function from noisy samples may be hard. Indeed, in full generality it may be impossible without exponentially many samples, since we may not see the same states across games. However, we may make this problem tractable by restricting χ to live in some class of functions Ω. For example, χ could be a linear function; this may approximate the truth when mapping *Spades* to *Hearts*, since having high cards is good in *Spades* but bad in *Hearts*. Ongoing work explores the feasibility of this approach, and we also plan to explore theoretical properties in future work.

References

[1] N. Brown and T. Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359:418–424, 2018.

[2] D. Silver et al. Mastering the game of Go without human knowledge. *Nature*, 550:354–359, 2017.

[3] V. Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[4] A. J. Hoane Jr. M. Campbell and F. Hsu. Deep blue. *Artificial Intelligence*, 134(1-2):57–83, 2002.

[5] M. Riedmiller. Neural fitted Q iteration - First experiences with a data efficient neural reinforcement learning method. *European conference on Machine Learning*, 05:317–328, 2005.

[6] G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

The detour problem in a stochastic environment: Tolman revisited

Pegah Fakhari
Department of psychological and brain sciences
Indiana University
Bloomington, IN
pfakhari@indiana.edu

Arash Khodadadi
Department of psychological and brain sciences
Indiana University
Bloomington, IN
arashkhoda@indiana.edu

Jerome R. Busemeyer
Department of psychological and brain sciences
Indiana University
Bloomington, IN
jbusemey@indiana.edu

Abstract

We designed a grid world task to study human planning and re-planning behavior in an unknown stochastic environment. In our grid world, participants were asked to travel from a random starting point to a random goal position while maximizing their reward. Because they were not familiar with the environment, they needed to learn its characteristics from experience to plan optimally. Later in the task, we randomly blocked the optimal path to investigate whether and how people adjust their original plans to find a detour. To this end, we developed and compared 12 different models. These models were different on how they learned and represented the environment and how they planned to catch the goal. The majority of our participants were able to plan optimally. We also showed that people were capable of revising their plans when an unexpected event occurred. The result from the model comparison showed that the model-based reinforcement learning approach provided the best account for the data and outperformed heuristics in explaining the behavioral data in the re-planning trials.

Keywords: planning and re-planning in stochastic environment, multistage decision making, navigation in maze, model-based reinforcement learning

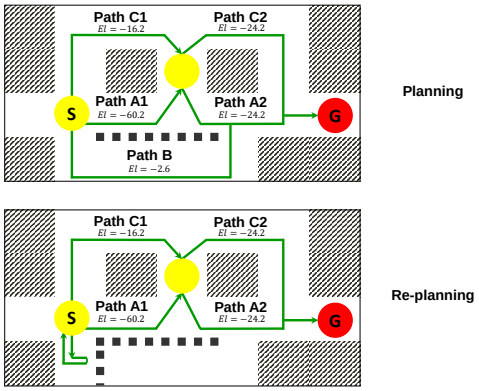


Figure 1: **General grid world used in our experiments**. The decision nodes are represented by yellow circles and the goal is depicted by red circle. The green paths are available to participants. Note that participants can not see the obstacles depicted by shadow areas. The black dashed line in the grid separates path B and path A1A2. Top: In the planning trials, for this current start and goal positions, 5 paths are available to participants and they need to find the optimal path. Down: In the re-planning trials, the optimal path, path **B** is blocked and participants need to find the detour.

1 Introduction

This material was published in cognitive psychology, (Fakhari, Khodadadi, & Busemeyer, 2017).

Planning in a stochastic environment is challenging. It becomes even more challenging when the environment is unknown to us. No matter how complicated these problems are, we mainly use our previous experiences to deal with them. Sometimes the environment changes and forces us to change or modify our plan. As a result, we update our plan every now and then to make sure our plan becomes a success. In this paper, we tried to study real life planning and re-planning problems in a simplified situation using our grid world experiment. Using this framework, we developed three experiments to investigate planning and re-planning in humans while learning an unknown environment.

2 Design

After 6 blocks of training in a 4 by 7 grid, participants' planning skill was tested, Fig. 1. At the beginning of the 7th block (the pretest block), we warned them that their score will be converted to monetary reward (the exchange rate was 0.01). In experiment 1 and 2, the starting and goal positions were fixed during the test phase, but, in experiment 3, we randomized these pairs. The majority of our participants (19 out of 19 in experiment 1, 30 out of 36 in experiment 2 and 30 out of 32 in experiment 3) were able to find the optimal path. This is in contrast to previous studies that showed people are more likely to be sub-optimal in the description-based decision trees involving a probabilistic reward, (Hey & Knoll, 2011), (Hotaling & Busemeyer, 2012)[1].

The basic configuration of the grid world is similar to the detour problem in (Tolman, Ritchie, & Kalish, 1946), but we modified it into a stochastic environment where finding the shortest path was not optimal anymore. It is important to note that the experiments had two distinctive features that encourage goal-directed behavior: first, the (hidden) punishments were probabilistic, and second, the starting and goal positions were randomly located in different cells. In order to find the optimal path, participants needed to learn and compare the expected values of different paths. Employing probabilistic rewards

[1]It has been proposed by (Erev, Ert, Plonsky, Cohen, & Cohen, 2017) that people show planning biases in description-based decision trees problems, but not in experience based learning

enabled us to represent this problem in a decision tree framework to study (optimal) planning similar to experience-based decision-making tasks.

Table 1 highlights the differences and similarities among the three experiments. It is important to emphasize that participants can only see a plain grid on the screen along with their current position (yellow circle) and their destination (red circle, Fig. 1) with no sign/cue of the obstacles or the stochastic losses. They do not have access to papers nor calculators/cellphones to do any computations or to take notes. In the instruction, they are told to consider a scenario that they move to a new city (a 4 by 7 grid) and need to get from one place to another (determined by the yellow and red circles). In the test phase, participants are told that a random accident might happen in one of the possible routes and block that path. If they see the accident they need to find a detour path.

Table 1: Summary Of Experiments

Design	Experiment 1	Experiment 2	Experiment 3
Number of learning blocks	6	6	6
Number of test blocks	2	2	3
Number of pretest blocks	1	1	1
Number of probabilistic losses	2	5	5
Fixed G and S in the test blocks	Yes	Yes	**No**
Fixed G and S in the learning blocks	No	No	No
Fixed G and S in the pretest block	No	No	No
Cells with stochastic losses	15, 16	9, 11, 16, 19, 17	9, 11, 16, 19, 17
Number of re-planning trials in the test trials	13 out of 40	13 out of 40	20 out of 60

Experiment 1 has two stochastic losses with one sure loss at cell 21 with the loss of -45. In experiment 2 and 3, there are 5 stochastic losses (similar environment). The rate of re-planning trials in the test blocks is fixed in all three experiments (33%). Unlike experiments 1 and 2, the pairs in the test blocks of the experiment 3 are random. In all three experiments, there is one pretest block and six learning blocks. G and S stand for Goal and Starting positions.

3 Model fit results

We developed 12 different models to fit the choice data, Table 2. These models were different on how they learned and represented the environment and how they planned to catch the goal. The baseline model selects a random action at each cell regardless of what participants have experienced. This model was the simplest model in our benchmark. The traditional Q-learning algorithm was not able to explain the data because the starting and goal positions in the environment in our experiments were not fixed. In addition, when a change happens in the environment, the whole set of the action-state values needs to be updated (again by extensive amount of learning and exposure to the new environment).

The (full) model-based RL tries to learn the model of the environment by estimating the transition and reward functions. This knowledge (of the environment) is later used to generate Q-value for each action. The model-based RL had the best predictions in the test blocks. In model 7, we restrict the model-based RL in its spatial search. Instead of a complete tree search that is commonly used in value iteration, we confine the model's planning depth to its kth nearest neighbors (k is a free parameter). While this modification can decrease the computational costs, for many pairs the best fitted k leads to a full tree search.

In addition to model-based and model-free RL, there is another alternative, SR, which is more flexible than model-free RL and computationally simpler than model-based RL, (Dayan, 1993). SR calculates the state values using both reward and a successor map which stores the expected and discounted future states' occupancies. In case of reward devaluation, SR's behavior is similar to model-based RL but when there is an alteration in the transition structure, it fails to adapt to the change (similar to model-free RL), (Gershman, Moore, Todd, Norman, & Sederberg, 2012), (Kulkarni, Saeedi, Gautam, & Gershman, 2016), (Momennejad et al., 2016).

Table 2: Summary of models and their free parameters

Model	Description	Parameters	No. of free parameters
1	Baseline (random model)	-	0
2	Q-learning	$\beta, \gamma, q_{u_0}, q_{r_0}, q_{d_0}, q_{l_0}$	6
3	Model-based RL	$\beta, \gamma, \alpha_{1_{HS}}, \alpha_{1_{MS}}, \alpha_{1_{LS}}$	5
4	Avoids Salient Loss -MBRL	β, γ	2
5	Remembers The Last R -MBRL	β, γ	2
6	Finds The Shortest Path -MBRL	β, γ	2
7	Cubed Model-Based RL	$\beta, \gamma, \alpha_{1_{HS}}, \alpha_{1_{MS}}, \alpha_{1_{LS}}, \omega$	6
8	Successor Representation	$\beta, \gamma, \alpha_l, \alpha_{1_{HS}}, \alpha_{1_{MS}}, \alpha_{1_{LS}}$	6
9	Avoids Salient Loss -SR	β, γ, λ	3
10	Remembers The Last R -SR	β, γ, λ	3
11	Finds The Shortest Path -SR	β, γ, λ	3
12	Hybrid SR-MB	$\beta, \gamma, \alpha_l, \omega_{hb}, \alpha_{1_{HS}}, \alpha_{1_{MS}}, \alpha_{1_{LS}}$	7

MB = Model-based, SR = Successor Representation.

Although the hybrid SR-MB model provided a better account for participants' choices in the learning blocks, it failed to predict the re-planning behavior in the test blocks. Since the candidate models were not trained by the test blocks' data, they needed to *generalize* their knowledge (from the learning blocks) to perform optimally in the re-planning trials when the optimal path was randomly blocked. One solution to this problem is to use a mixture model which switches from the SR model in the planning trials to the model-based RL mechanism in the re-planning trials. It can post hoc fit all the data the best, because it uses the best fitting model for the training blocks and then switches to the best predicting model for the test blocks.

References

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, *5*(4), 613–624.

Erev, I., Ert, E., Plonsky, O., Cohen, D., & Cohen, O. (2017). From anomalies to forecasts: Toward a descriptive model of decisions under risk, under ambiguity, and from experience. *Psychological Review*, *124*(4), 369–409.

Fakhari, P., Khodadadi, A., & Busemeyer, J. (2017, September). The detour problem in a stochastic environment: Tolman revisited. *arXiv:1709.09761 [q-bio, stat]*.

Gershman, S. J., Moore, C. D., Todd, M. T., Norman, K. A., & Sederberg, P. B. (2012). The successor representation and temporal context. *Neural Computation*, *24*(6), 1553–1568.

Hey, J. D., & Knoll, J. A. (2011). Strategies in dynamic decision makingAn experimental investigation of the rationality of decision behaviour. *Journal of Economic Psychology*, *32*(3), 399–409.

Hotaling, J. M., & Busemeyer, J. R. (2012). DFT-D: a cognitive-dynamical model of dynamic decision making. *Synthese*, *189*(1), 67–80.

Kulkarni, T. D., Saeedi, A., Gautam, S., & Gershman, S. J. (2016). Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Momennejad, I., Russek, E. M., Cheong, J. H., Botvinick, M. M., Daw, N., & Gershman, S. J. (2016). The successor representation in human reinforcement learning. *bioRxiv*.

Tolman, E. C., Ritchie, B., & Kalish, D. (1946). Studies in spatial learning. II. Place learning versus response learning. *Journal of Experimental Psychology*, *36*(3), 221–229.

Generalization and Regularization in DQN *

Jesse Farebrother
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
jfarebro@ualberta.ca

Marlos C. Machado †
Google Brain
Montréal, QC, Canada
marlosm@google.com

Michael Bowling
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
mbowling@ualberta.ca

Abstract

Deep reinforcement learning (RL) algorithms have shown an impressive ability to learn complex control policies in high-dimensional environments. However, despite the ever-increasing performance on popular benchmarks such as the Arcade Learning Environment (ALE), policies learned by deep RL algorithms often struggle to generalize when evaluated in remarkably similar environments. In this paper, we assess the generalization capabilities of DQN, one of the most traditional deep RL algorithms in the field. Additionally, we provide evidence suggesting that DQN overspecializes to the training environment. Furthermore, we comprehensively evaluate the impact of traditional regularization methods, ℓ_2-regularization and dropout, and of reusing the learned representations to improve the generalization capabilities of DQN. We perform this study using different game modes of Atari 2600 games, a recently introduced modification for the ALE which supports slight variations of the Atari 2600 games traditionally used for benchmarking. Despite regularization being largely underutilized in deep RL, we show that it can, in fact, help DQN learn more general features. These features can then be reused and fine-tuned on similar tasks, considerably improving the sample efficiency of DQN.

Keywords: Deep reinforcement learning, generalization, regularization, representation learning, function approximation, Atari games

Acknowledgements

The authors would like to thank Matthew E. Taylor, Tom Van de Wiele, and Marc G. Bellemare for useful discussions, as well as Vlad Mnih for feedback on a preliminary draft of the manuscript. This work was supported by funding from NSERC and Alberta Innovates Technology Futures through the Alberta Machine Intelligence Institute (Amii). Computing resources were provided by Compute Canada through CalculQuébec.

*This paper is an extended abstract of the following article: "J. Farebrother, M. C. Machado, M. Bowling. Generalization and Regularization in DQN. CoRR abs/1810.00123, 2018."

†Work performed while at the Department of Computing Science at the University of Alberta.

1 Introduction

Recently, reinforcement learning (RL) has proven very successful on complex high-dimensional problems, in large part due to the increase in computational power and to the use of deep neural networks for function approximation [e.g., 4]. Despite the generality of the proposed solutions, applying these algorithms to slightly different environments generally requires agents to learn the new task from scratch. On the other hand, deep neural networks are lauded for their generalization capabilities [e.g., 2] with some communities heavily reusing the learned representations [e.g., 5]. In light of the successes of traditional supervised learning methods, the current lack of generalization or reusable knowledge (i.e., policies, representation) acquired by current deep RL algorithms is somewhat surprising.

In this paper we investigate whether the representations learned by deep RL methods can be generalized, or at the very least reused and refined on small variations to the task at hand. We evaluate the generalization capabilities of DQN [4] and we further explore whether the experience gained by the supervised learning community to improve generalization and to avoid overfitting could be used in deep RL. We employ conventional supervised learning techniques, albeit largely unexplored in deep RL, such as fine-tuning (i.e., reusing and refining the representation) and regularization. In order to perform this study we employ different game modes and difficulties of Atari 2600 games, a feature recently introduced to the Arcade Learning Environment (ALE) [1, 3].

We show that a learned representation trained with regularization allows us to learn more general features capable of being reused and fine-tuned. Besides improving the generalization capabilities of the learned policies this fine-tuning procedure has the potential to greatly improve sample efficiency on settings in which an agent might face multiple variations of the same task, or when future tasks are unknown to the agent. Our results suggest that, if we move beyond the single-task setting in RL, regularization techniques might play a prominent role in deep RL algorithms.

2 Background

2.1 Reinforcement learning

Q-learning [7] is a traditional approach to learning an optimal state-action value function from samples obtained by interacting with the environment. The agent updates the state-action value function iteratively using the update rule

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} \quad + \gamma \max_{a' \in \mathcal{A}} Q(S_{t+1}, a') - Q(S_t, A_t)\big]. \tag{1}$$

Generally, due to the exploding size of the state space in many real-world problems, it is intractable to learn a state-action pairing for the entire MDP, we often resort to learning an approximation to q_π.

DQN approximates the state-action value function such that $Q(s, a;\, \theta) \approx q_\pi(s, a)$, where θ denotes the weights of a neural network. DQN is trained in a supervised fashion to minimize the squared TD error from 1 using a batch of experience sampled uniformly from a buffer of $(S_t, A_t, R_{t+1}, S_{t+1})$ transition tuples.

2.2 Supervised learning

In the **supervised learning** problem we wish to learn a function $f : \mathbb{R}^n \to \mathbb{R}$ which maps training examples X_i of dimension n to a predicted output label $\hat{y}_i$. The model should accurately predict true label y_i while generalizing to unseen examples. When f is a neural network parametrized by θ we typically train the model by minimizing the objective

$$\min_{\theta} \ \frac{\lambda}{2} \ \|\theta\|_2^2 + \frac{1}{m} \sum_{i=1}^{m} L(y_i, f(X_i;\, \theta)), \tag{2}$$

where L is a differentiable loss function which outputs a scalar corresponding to the quality of the prediction (e.g., squared error loss). The first term in 2 is a form of regularization, i.e., $\boldsymbol{\ell_2}$ **regularization**, which encourages generalization by imposing a penalty on large weight vectors. The hyperparameter λ is the weighted importance of the regularization term.

Dropout [6] is a regularization technique applied to the feed-forward operation of a neural network. During the forward pass each neural unit has a chance of being set to 0 according to $\mathrm{Bernoulli}(p)$ where p is referred to as the dropout rate.

Fine-tuning is the process of bootstrapping weight initialization using pre-trained weights from a different task. This differes from standard stochastic initialization processes.

3 The ALE as a platform for evaluating generalization in RL

The Arcade Learning Environment (ALE) is a platform used to evaluate agents across dozens of Atari 2600 games [1]. The ALE poses the problem of general competency by having agents use the same learning algorithm to perform well in as many games as possible without using game specific knowledge.

Table 1: Direct policy evaluation. Each agent is initially trained in the default flavour for 50M frames then evaluated in each listed game flavour. Agents with regularization are trained using dropout and ℓ_2 regularization. Reported numbers are averaged over five runs. Standard deviation is reported between parentheses.

Game Variant		Eval. with Regularization		Eval. without Regularization		Learn Scratch	
Freeway	m1d0	**5.8**	**(3.5)**	0.2	(0.2)	4.8	(9.3)
	m1d1	**4.4**	**(2.3)**	0.1	(0.1)	0.0	(0.0)
	m4d0	20.6	(0.7)	15.8	(1.0)	**29.9**	**(0.7)**
Hero	m1d0	116.8	(76.0)	82.1	(89.3)	**1425.2**	**(1755.1)**
	m2d0	30.0	(36.7)	33.9	(38.7)	**326.1**	**(130.4)**
Breakout	m12d0	31.0	(8.6)	43.4	(11.1)	**67.6**	**(32.4)**
Space Invaders	m1d0	456.0	(221.4)	258.9	(88.3)	**753.6**	**(31.6)**
	m1d1	146.0	(84.5)	140.4	(61.4)	**698.5**	**(31.3)**
	m9d0	290.0	(257.8)	179.0	(75.1)	**518.0**	**(16.7)**

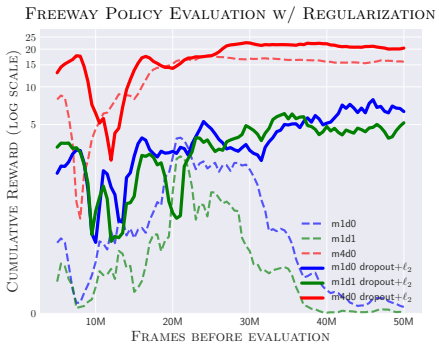


Figure 1: Evaluation performance in different flavours of Freeway from a trained agent in m0d0 with regularization (solid line) and without regularization (dashed line). Curves are smoothed using a moving average over two data points. Results were averaged over five seeds.

In this paper, we use the different modes and difficulties of Atari 2600 games to evaluate a neural network's ability to generalize in high-dimensional state spaces. Game modes, originally native to the Atari 2600 console, were recently added in the ALE [3]. They give us modifications of the default environment dynamics and state space, often modifying sprites, velocities, and partial observability. These modes pose a tractable way to investigate generalization of RL agents in a high-dimensional environment.

In this paper, we use 13 flavours (combinations of a mode and a difficulty) obtained from 4 games: Freeway, Hero, Breakout, and Space Invaders *. In Freeway, modes vary the speed and number of vehicles, while different difficulties change how the player is penalized for running into a vehicle. In Hero, subsequent modes start the player off at increasingly harder levels of the game. The mode we use in Breakout makes the bricks partially observable. Modes of Space Invaders allow for oscillating barriers, increasing the width of the player sprite, and partially observable aliens.

4 Generalization of the learned policies and overfitting

In order to test the generalization capabilities of DQN we first evaluate whether the policy learned after 50M frames in the default flavour, m0d0 (mode 0, difficulty 0) can perform well in a different flavour. If the representation encodes a robust policy we might expect it to be able to generalize to slight variations of the underlying reward signal, game dynamics, or observations. We measure the cumulative reward averaged over 100 episodes in the new flavour. The results are summarized in Table 1 under the column Eval. Without Regularization. Baseline results where the agent is trained from scratch in the target flavour used for evaluation are summarized in the baseline column Learn Scratch.

We can see in the results that the policies learned by DQN do not generalize well to different flavours, even when the flavours are remarkably similar. For example, in Freeway, a high-level policy applicable to all flavours is to go up while avoiding cars. The default flavour m0d0 and m4d0 comprise of exactly the same sprites, the only difference being that in m4d0 some cars accelerate and decelerate over time. The close to optimal policy learned in m0d0 is only able to score around half of what is achieved by the policy learned from scratch in m4d0.

As aforementioned, the different modes of Hero can be seen as giving the agent a curriculum. Interestingly, the agent trained in the default mode for 50M frames can progress to at least level 3 and sometimes level 4. Mode 1 starts the agent off at level 5, and performance in this mode suffers greatly during evaluation despite no addition of any novel game mechanics. Perhaps the agent is memorizing trajectories instead of learning a robust policy capable of solving each level.

Results in some flavours suggest that the agent is overfitting to the flavour it is trained on. We tested this hypothesis by periodically evaluating the policy being learned in each other flavour of that game. This process involved taking checkpoints of the network every 500,000 frames and evaluating the policy in the prescribed flavour. The results obtained in Freeway, the most pronounced game in which we observe overfitting, are depicted by dashed lines in Figure 1.

In Freeway, while we see the policy's performance flattening out in m4d0, we do see the traditional bell-shaped curve associated to overfitting in the other modes. At first, improvements in the original policy do correspond to improvements

*Video comparing each game flavour can be found at https://goo.gl/pCvPiD

in the performance of that policy in other flavours. With time, it seems that the agent starts to refine its policy for the specific flavour it is being trained on, overfitting to that flavour. With other game flavours being significantly more complex in their dynamics and gameplay, we do not observe this prominent bell-shaped curve though. For example, in BREAKOUT, we actually observe a monotonic increase in performance throughout the evaluation process.

5 Regularization in deep RL

In order to evaluate the hypothesis that the observed lack of generalization is due to overfitting, we revisit some popular regularization methods from the supervised learning literature. We consider two forms of regularization: dropout and ℓ_2 regularization.

First we want to understand the effect of regularization when deploying the learned policy in a different flavour. We do so by applying dropout to the first four layers of the network, that is, the three convolutional layers and the first fully connected layer. Note that we used two seperate dropout rates, one for the convolutional layers and the other for the fully connected layer. We also evaluate the use ℓ_2 regularization on all weights in the network during training. An exhaustive grid search was performed for both dropout and ℓ_2 regularization and we ended up combining both methods as this provided a good balance between training and evaluation performance. For all future experiments we fixed $\lambda = 10^{-4}$, and $p_{\text{conv}}, p_{\text{fc}} = 0.05, 0.1$. We follow the same evaluation methodology described when examining the non-regularized policy.

Alongside the column EVAL. WITHOUT REGULARIZATION in Table 1 we present the results for EVAL. WITH REGULARIZATION. In most flavours, evaluating the policy trained with regularization does not negatively impact performance when compared to the performance of the policy trained without regularization. In some flavours we even see an increase in performance. Interestingly, when using regularization the agent in FREEWAY improves for all flavours and even learns a policy capable of outperforming the baseline learned from scratch in two of the three flavours. Moreover, in FREEWAY we now observe increasing performance during evaluation throughout most of the learning procedure as depicted with solid lines in Figure 1. The results in Figure 1 seem to confirm the notion of overfitting.

Despite slight improvements from these techniques, regularization by itself does not seem sufficient to enable policies to generalize across flavours. As shown in the next section, the real benefit of regularization in deep RL seems to come from the ability to learn more general features. These features may lead to a more adaptable representation which can be reused and subsequently fine-tuned on other flavours, which is often the case in supervised learning.

6 Value function fine-tuning

We hypothesize that the benefit of regularizing deep RL algorithms may not come from improvements during evaluation, but instead in having a good parameter initialization that can be adapted to new tasks that are similar. We evaluate this hypothesis by fine-tuning the entire network post-training with and without regularization.

When fine-tuning the entire network we take the weights of the network trained in the default flavour for 50M frames and use them to initialize the network commencing training in the new flavour for 50M frames. For comparison we provide a baseline trained from scratch for 50M and 100M frames in each flavour. Directly comparing the performance obtained after fine-tuning to the performance after 50M frames (SCRATCH) shows the benefit of re-using a learned representation. Comparing the performance obtained after fine-tuning to the performance of 100M frames (SCRATCH) lets us take into consideration the sample efficiency of the whole learning process. The results are presented in Table 2.

Fine-tuning from a *non-regularized representation* yields conflicting conclusions. Although in FREEWAY we obtained positive fine-tuning results, we note that rewards are so sparse in mode 1 that this initialization is likely to be acting as a form of optimistic initialization, biasing the agent to go up. The agent observes rewards more often, therefore, it learns quicker about the new flavour. However, the agent is still unable to reach the maximum score in these flavours.

The results of fine-tuning the *regularized representation* are more exciting. In FREEWAY we observe the highest scores on m1d0 and m1d1 throughout the whole paper. In HERO we vastly outperform fine-tuning from an non-regularized representation. In SPACE INVADERS we obtain higher scores across the board on average when comparing to the same amount of experience. These results suggest that reusing a regularized representation in deep RL might allow us to learn more general features which can be more successfully fine-tuned.

Moreover, initializing the network with a regularized representation has a big impact on the agent's performance when compared to initializing the network randomly. These results are impressive when we consider the potential regularization has in reducing the sample complexity of deep RL algorithms. Such an observation also holds when we take the total number of frames seen between two flavours into consideration. When directly comparing one row of REGULARIZED FINE-TUNING to SCRATCH we are comparing two algorithms that observed 100M frames. However, to generate two rows of SCRATCH we used 200M frames while two rows of REGULARIZED FINE-TUNING used 150M frames (50M from scratch + 50M in each row). The distinction becomes larger as more tasks are taken into consideration.

Table 2: Experiments fine-tuning the entire network with and without regularization (dropout + ℓ_2). An agent is trained with dropout + ℓ_2 regularization in the default flavour of each game for 50M frames, then DQN's parameters were used to initialize the fine-tuning procedure on each new flavour for 50M frames. The baseline agent is trained from scratch up to 100M frames. Standard deviation is reported between parentheses.

		FINE-TUNING				REGULARIZED FINE-TUNING				SCRATCH			
GAME VARIANT		10M		50M		10M		50M		50M		100M	
FREEWAY	m1d0	2.9	(3.7)	22.5	(7.5)	20.2	(1.9)	**25.4**	**(0.2)**	4.8	(9.3)	7.5	(11.5)
	m1d1	0.1	(0.2)	17.4	(11.4)	18.5	(2.8)	**25.4**	**(0.4)**	0.0	(0.0)	2.5	(7.3)
	m4d0	20.8	(1.1)	31.4	(0.5)	22.6	(0.7)	32.2	(0.5)	29.9	(0.7)	**32.8**	**(0.2)**
HERO	m1d0	220.7	(98.2)	496.7	(362.8)	322.5	(39.3)	4104.6	(2192.8)	1425.2	(1755.1)	**5026.8**	**(2174.6)**
	m2d0	74.4	(31.7)	92.5	(26.2)	84.8	(56.1)	211.0	(100.6)	326.1	(130.4)	**323.5**	**(76.4)**
BREAKOUT	m12d0	11.5	(10.7)	69.1	(14.9)	48.2	(4.1)	**96.1**	**(11.2)**	67.6	(32.4)	55.2	(37.2)
SPACE INVADERS	m1d0	617.8	(55.9)	926.1	(56.6)	701.8	(28.5)	**1033.5**	**(89.7)**	753.6	(31.6)	979.7	(39.8)
	m1d1	482.6	(63.4)	799.4	(52.5)	656.7	(25.5)	**920.0**	**(83.5)**	698.5	(31.3)	906.9	(56.5)
	m9d0	354.8	(59.4)	574.1	(37.0)	519.0	(31.1)	**583.0**	**(17.5)**	518.0	(16.7)	567.7	(40.1)

7 Discussion and conclusion

Analyzing generalization and overfitting in deep RL has its own issues on top of the challenges posed in the supervised learning case. Actually, generalization in RL can be seen in different ways. We can talk about generalization in RL in terms of conditioned sub-goals within an environment, learning multiple tasks at once, or sequential task learning as in a continual learning setting. In this paper we evaluated generalization in terms of small variations of high-dimensional control tasks. This provides a candid evaluation method to study how well representations learned by deep neural networks in RL problems can generalize. The approach of studying generalization with respect to the representation learning problem intersects nicely with the aforementioned problems in RL where generalization is key.

Ultimately we want agents that can keep learning as they interact with the world in a continual learning fashion. The ability to generalize is essential. Throughout this paper we often avoided the expression *transfer learning* because we believe that succeeding in slightly different environments should be actually seen as a problem of generalization. Our results suggested that regularizing and fine-tuning representations in deep RL might be a viable approach towards improving sample efficiency and generalization in this setting.

References

[1] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 512–519, 2014.

[6] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[7] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Hyperbolic Discounting and Learning over Multiple Horizons

William Fedus
Google Brain
University of Montreal (Mila)
liamfedus@google.com

Carles Gelada
Google Brain
cgel@google.com

Yoshua Bengio
University of Montreal (Mila)
CIFAR Senior Fellow
yoshua.bengio@mila.quebec

Marc G. Bellemare
Google Brain
bellemare@google.com

Hugo Larochelle
Google Brain
hugolarochelle@google.com

Abstract

Reinforcement learning (RL) typically defines a discount factor (γ) as part of the Markov Decision Process. The discount factor values future rewards by an exponential scheme that leads to theoretical convergence guarantees of the Bellman equation. However, evidence from psychology, economics and neuroscience suggests that humans and animals instead have *hyperbolic* time-preferences ($\frac{1}{1+kt}$ for $k > 0$). Here we extend earlier work of Kurth-Nelson and Redish and propose an efficient *deep reinforcement learning* agent that acts via hyperbolic discounting and other non-exponential discount mechanisms. We demonstrate that a simple approach approximates hyperbolic discount functions while still using familiar temporal-difference learning techniques in RL. Additionally, and independent of hyperbolic discounting, we make a surprising discovery that simultaneously learning value functions over multiple time-horizons is an effective auxiliary task which often improves over a strong value-based RL agent, Rainbow.

Keywords: Reinforcement learning (RL), time-preferences, hyperbolic discounting, multiple-horizon discounting, auxiliary tasks

Acknowledgements

This research and its general framing drew upon the talents of many researchers at Google Brain, DeepMind and Mila. In particular, we'd like thank Ryan Sepassi for framing of the paper, Utku Evci for last minute Matplotlib help, Audrey Durand, Margaret Li, Adrien Ali Taïga, Ofir Nachum, Doina Precup, Jacob Buckman, Marcin Moczulski, Nicolas Le Roux, Ben Eysenbach, Sherjil Ozair, Anirudh Goyal, Ryan Lowe, Robert Dadashi, Chelsea Finn, Sergey Levine, Graham Taylor and Irwan Bello for general discussions and revisions.

1 Introduction

The standard treatment of the reinforcement learning (RL) problem is the Markov Decision Process (MDP) which includes a discount factor $0 \leq \gamma < 1$ that exponentially reduces the present value of future rewards [18]. A reward r_t received in t-time steps is devalued to $d(t)r_t = \gamma^t r_t$, a discounted utility model. This establishes a time-preference for rewards realized sooner rather than later. The decision to exponentially discount future rewards by γ leads to value functions that satisfy theoretical convergence properties [3]. The magnitude of γ also plays a role in stabilizing learning dynamics of RL algorithms [3] and is often treated as a hyperparameter.

However, discounting future values exponentially and according to a single discount factor γ does not harmonize with the measured value preferences in humans and animals [10]. A wealth of empirical evidence has been amassed that humans, monkeys, rats and pigeons instead discount future returns *hyperbolically*, where $d_k(t) = \frac{1}{1+kt}$, for some positive $k > 0$ [1, 12, 6]. As an example of a potential difference emerging from hyperbolic discounting, one may maintain time-inconsistent preferences. For instance, a person might prefer \$1M immediately to the promise of \$1.1M dollars tomorrow while simultaneously preferring the promise of \$1.1M in 366 days over \$1M in 365 days.

Hyperbolic time-preferences is mathematically consistent with the agent maintaining some uncertainty over the prior belief of the *hazard rate* in the environment [16]. Hazard rate $h(t)$ measures the per-time-step risk the agent incurs as it acts in the environment due to a potential early death. Precisely, if $s(t)$ is the probability that the agent is alive at time t then the hazard rate is $h(t) = -\frac{d}{dt}\ln s(t)$. We consider the case where there is a fixed, but potentially unknown hazard rate $h(t) = \lambda \geq 0$. The prior belief of the hazard rate $p(\lambda)$ implies a specific discount function [16]. Under this formalism, the canonical case in RL of discounting future rewards according to $d(t) = \gamma^t$ is consistent with the belief that there exists a single hazard rate $\lambda = e^{-\gamma}$ known with certainty. This contrasts most RL environments with simple hazard dynamics.

We extend the μAgents work of Kurth-Nelson and Redish which empirically demonstrated modeling a finite set of exponential functions may approximate a hyperbolic discounting function [9] which is a model consistent with fMRI studies [19, 14]. Here we propose a deep reinforcement learning agent that efficiently approximates hyperbolic discounting while preserving Q-learning [20] tools and their associated theoretical guarantees. Our agent efficiently models many Q-values in parallel by building separate Q-value heads off a common shared representation of the state.

Surprisingly and in addition to enabling new discounting schemes, we observe that learning a set of Q-values is beneficial as an auxiliary task [8]. Adding this *multi-horizon auxiliary task* often improves over strong baselines including Rainbow [7] in the ALE [2]. This work questions the RL paradigm of learning policies through a single discount function which exponentially discounts future rewards through two contributions:

1. **Hyperbolic-agent.** A practical approach for training an efficient deep RL agent which discounts future rewards by a hyperbolic discount function and acts according to this.
2. **Multi-horizon auxiliary task.** A demonstration of multi-horizon learning over many γ simultaneously as an effective auxiliary task in deep RL.

2 Hazard in MDPs

To study MDPs with *hazard distributions* and *general discount functions* we introduce two modifications. The hazardous MDP now is defined by the tuple $< \mathcal{S}, \mathcal{A}, R, P, \mathcal{H}, d >$. In standard form, the state space $\mathcal{S}$ and the action space $\mathcal{A}$ may be discrete or continuous. The learner observes samples from the environment transition probability $P(s_{t+1}|s_t, a_t)$ for going from $s_t \in \mathcal{S}$ to $s_{t+1} \in \mathcal{S}$ given $a_t \in \mathcal{A}$. We will consider the case where P is a sub-stochastic transition function, which defines an episodic MDP. The environment emits a bounded reward $r : \mathcal{S} \times \mathcal{A} \to [r_{min}, r_{max}]$ on each transition. In this work we consider non-infinite episodic MDPs. A policy $\pi : \mathcal{S} \to \mathcal{A}$ is a mapping from states to actions.

The first difference is that at the beginning of each episode, a hazard $\lambda \in [0, \infty)$ is sampled from the hazard distribution $\mathcal{H}$. This is equivalent to sampling a *continuing* probability $\gamma = e^{-\lambda}$. During the episode, the hazard modified transition function will be P_λ, in that $P_\lambda(s'|s, a) = e^{-\lambda}P(s'|s, a)$. The second difference is that we now consider a general discount function $d(t)$. This differs from the standard approach of exponential discounting in RL with γ according to $d(t) = \gamma^t$, which is a special case. The state action value function $Q_\pi^{\mathcal{H},d}(s, a)$ is the expected discounted rewards after taking action a in state s and then following policy π until termination.

$$Q_\pi^{\mathcal{H},d}(s, a) = \mathbb{E}_\lambda \mathbb{E}_{\pi,P_\lambda}\left[\sum_{t=0}^{\infty} d(t)R(s_t, a_t)|s_0 = s, a_0 = a\right] \tag{1}$$

where $\lambda \sim \mathcal{H}$ and $\mathbb{E}_{\pi,P_\lambda}$ implies that $s_{t+1} \sim P_\lambda(\cdot|s_t, a_t)$ and $a_t \sim \pi(\cdot|s_t)$. This formulation supports an equivalence between the value function of an MDP with a discount function and MDP with a hazard distribution.

3 Computing Hyperbolic Q-Values From Exponential Q-Values

We can re-purpose exponentially-discounted Q-values to compute hyperbolic (and other-non-exponential) discounted Q-values. The central challenge with using non-exponential discount strategies is that most RL algorithms use some form of TD learning [17]. This family of algorithms exploits the Bellman equation which, when using exponential discounting, relates the value function at one state with the value at the following state $Q_\pi^{\gamma^t}(s,a) = \mathbb{E}_{\pi,P}[R(s,a) + \gamma Q_\pi(s',a')]$ where expectation $\mathbb{E}_{\pi,P}$ denotes sampling $a \sim \pi(\cdot|s)$, $s' \sim P(\cdot|s,a)$, and $a' \sim \pi(\cdot|s')$.

Being able to reuse the literature on TD methods without being constrained to exponential discounting is thus an important challenge. By using a simple change of variables $\lambda = e^{-\gamma}$ relating hazard rate in Sozou's formulation [16] to discount rate in RL we may model hyperbolic as well as other non-exponential discount schemes through simple integrals over γ.

Lemma 3.1. *Let $Q_\pi^{\mathcal{H},\gamma}(s,a)$ be the state action value function under exponential discounting in a hazardous MDP $< \mathcal{S}, \mathcal{A}, R, P, \mathcal{H}, \gamma^t >$ and let $Q_\pi^{\mathcal{H},d}(s,a)$ refer to the value function in the same MDP except for new discounting $< \mathcal{S}, \mathcal{A}, R, P, \mathcal{H}, d >$. If there exists a function $w : [0,1] \to \mathbb{R}$ such that*

$$d(t) = \int_0^1 w(\gamma)\gamma^t d\gamma \tag{2}$$

which we will refer to as the exponential weighting condition, then

$$Q_\pi^{\mathcal{H},d}(s,a) = \int_0^1 w(\gamma) Q_\pi^{\mathcal{H},\gamma}(s,a) d\gamma \tag{3}$$

For example, the hyperbolic discount $d(t) = \frac{1}{1+kt}$ can be expressed as the integral $\frac{1}{k}\int_{\gamma=0}^1 \gamma^{1/k+t-1} d\gamma = \frac{1}{1+kt}$ where $\gamma = [0,1)$. This integral can be derived and the $w(\gamma) = \frac{1}{k}\gamma^{1/k-1}$ identified via the change of variables $\gamma = e^{-\lambda}$ applied to Sozou's Laplace transform [16] of the prior $\mathcal{H} = p(\lambda)$. This procedure allows us to express alternative discount factors in reinforcement learning.

4 Approximating the Discount Factor Integral

We now present a *practical* approach to approximate discounting $\Gamma(t) = \frac{1}{1+kt}$ using standard Q-learning [20]. To avoid estimating an infinite number of Q_π^γ-values we introduce a free hyperparameter (n_γ) which is the total number of Q_π^γ-values to consider, each with their own γ. We choose a practically-minded approach to choose $\mathcal{G} = [\gamma_0, \gamma_1, \cdots, \gamma_{n_\gamma}]$ that emphasizes evaluating larger values of γ rather than uniformly choosing points and empirically performs well as seen in Sections 5, 6.

Each $Q_\pi^{\gamma_i}$ computes the discounted sum of returns according to that specific discount factor $Q_\pi^{\gamma_i}(s,a) = \mathbb{E}_\pi[\sum_t (\gamma_i)^t r_t | s_0 = s, a_0 = a]$. The set of Q-values permits us to estimate the integral through a Riemann sum (Equation 5).

$$Q_\pi^\Gamma(s,a) = \int_0^1 w(\gamma) Q_\pi^\gamma(s,a) d\gamma \tag{4}$$

$$\approx \sum_{\gamma_i \in \mathcal{G}} (\gamma_{i+1} - \gamma_i)\ w(\gamma_i)\ Q_\pi^{\gamma_i}(s,a) \tag{5}$$

where we estimate the integral through a lower bound. This approach is similar to that of [9] where each μAgent models a specific discount factor γ. However, this differs in that our final agent computes a weighted average over each Q-value rather than a sampling operation of each agent based on a γ-distribution.

5 Pathworld Results

We first test our approach in a synthetic environment, Pathworld. The agent makes one decision in Pathworld: which of the N paths to investigate. Once a path is chosen, the agent continues until it reaches the end or until it dies. This is similar to a multi-armed bandit, with each action subject to dynamic risk. The paths vary quadratically in length with the index $d(i) = i^2$ but the rewards increase linearly with the path index $r(i) = i$, presenting a non-trivial decision. At deployment, an unobserved hazard $\lambda \sim \mathcal{H}$ is drawn and the agent is subject to a per-time-step risk of dying of $(1 - e^{-\lambda})$. Note that this takes inspiration from hazardous settings [16] since we determine time-preferences through risk to the reward. However, alternative formulations investigate these time-preferences emerging when rewards are subject to variable-timing as is the case in adjusting-delay procedure [11, 9].

Figure 1 confirms this approximates the true hyperbolic value of each path when the hazard prior matches the true distribution. Agents that discount exponentially according to a single γ (as is common in RL) incorrectly value the paths. Through this procedure, we are able to train an RL agent that is *robust* to hazards in the environment.

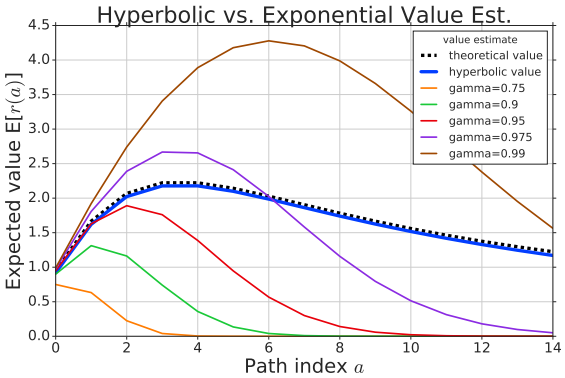


Figure 1: An unobserved hazard is drawn each episode $\lambda \sim p(\lambda)$ and the agent is subject to a per-step risk of the reward not being realized. When the agent's hazard prior matches the true hazard distribution, the value estimate agrees well with the theoretical value. Exponential discounts for many γ fail to well-approximate the true value as seen to the right in Table 1.

Discount function	MSE
hyperbolic value	**0.002**
γ=0.975	0.566
γ=0.95	1.461
γ=0.9	2.253
γ=0.99	2.288
γ=0.75	2.809

Table 1: The average mean squared error (MSE) over each of the paths in Figure 1 showing that our approximation scheme well-approximates the true value-profile.

We examine further the failure of exponential discounting in this hazardous setting. For this environment, the true hazard parameter in the prior was $k = 0.05$ (i.e. $\lambda \sim 20\exp(-\lambda/0.05)$). Therefore, at deployment, the agent must deal with dynamic levels of risk and faces a non-trivial decision of which path to follow. Even if we tune an agent's $\gamma = 0.975$ such that it chooses the correct arg-max path, it still fails to capture the functional form (Figure 1) and it achieves a high error over many paths (Table 1). If the arg-max action was not available or if the agent was proposed to evaluate non-trivial intertemporal decisions, it would act sub-optimally.

6 Atari 2600 Results

With our approach validated in Pathworld, we now move to the high-dimensional environment of Atari 2600, specifically, ALE [2]. We use the Rainbow variant from Dopamine [4] which implements three of the six considered improvements from the original paper: distributional RL, predicting n-step returns and prioritized replay buffers.

The agent maintains a shared hidden representation $h(s)$ of state, but computes n_γ separate Q-value logits (one for each of the γ_i) via $Q_\pi^{(i)}(s,a) = W_i h(s) + b_i$ where W_i and b_i are the learnable parameters of the affine transformation for that logit. On a random subset of 19 games from ALE we find performance improvements [5] of the Hyperbolic-Rainbow agent (Hyper-Rainbow), however, to dissect these improvements, recognize that Hyper-Rainbow changes two properties from the base Rainbow agent. First, the agent acts according to hyperbolic Q-values computed by our approximation described above. Second, the agent simultaneously learns Q-values over many γ rather than a Q-value for a single γ.

The second modification can be regarded as introducing an *auxiliary task* [8]. Therefore, to attribute the performance of each properly we construct a Rainbow agent augmented with the multi-horizon auxiliary task but have it still act according to the original policy (Multi-Rainbow). That is, Multi-Rainbow acts to maximize expected rewards discounted by a fixed γ_{action} but now learns over multiple horizons.

We examine further and investigate the performance of this auxiliary task across the full Arcade Learning Environment. Doing so we find strong empirical benefits of the multi-horizon auxiliary task on the Rainbow agent as shown in Figure 2. This provides a clear demonstration of the usefulness of modeling multiple time-scales as was sought by earlier γ-net work [15] and an alternative motivation from TD(Δ) [13].

7 Discussion

This work challenges one of the basic premises of RL: that the agent should always maximize the *exponentially discounted* returns via a *single* discount factor. Our deep RL agent instead builds a shared representation that efficiently learns over

multiple horizons simultaneously. This permits us both a broader class of learning algorithms and this also improves performance as a powerful auxiliary task that positively builds upon other improvements (distributional learning, n-step returns) previously introduced. Many open questions remain including which time-preferences are beneficial in more complicated environments and the general mechanism underlying the improvement of our auxiliary task and we hope these findings and other related work piques additional inquiry.

References

[1] G. Ainslie. Specious reward: a behavioral theory of impulsiveness and impulse control. *Psychological bulletin*, 82(4):463, 1975.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

[4] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare. Dopamine: A research framework for deep reinforcement learning. *CoRR*, abs/1812.06110, 2018.

[5] W. Fedus, C. Gelada, Y. Bengio, M. G. Bellemare, and H. Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

[6] L. Green, E. B. Fisher, S. Perlow, and L. Sherman. Preference reversal and self control: Choice as a function of reward amount and delay. *Behaviour Analysis Letters*, 1981.

[7] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[8] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[9] Z. Kurth-Nelson and A. D. Redish. Temporal-difference reinforcement learning with distributed representations. *PLoS One*, 4(10):e7362, 2009.

[10] T. V. Maia. Reinforcement learning, conditioning, and the brain: Successes and challenges. *Cognitive, Affective, & Behavioral Neuroscience*, 9(4):343–364, 2009.

[11] J. E. Mazur. An adjusting procedure for studying delayed reinforcement. 1987.

[12] J. E. Mazur. Choice, delay, probability, and conditioned reinforcement. *Animal Learning & Behavior*, 25(2):131–147, 1997.

[13] J. Romoff, P. Henderson, A. Touati, Y. Ollivier, E. Brunskill, and J. Pineau. Separating value functions across time-scales. *arXiv preprint arXiv:1902.01883*, 2019.

[14] N. Schweighofer, M. Bertin, K. Shishida, Y. Okamoto, S. C. Tanaka, S. Yamawaki, and K. Doya. Low-serotonin levels increase delayed reward discounting in humans. *Journal of Neuroscience*, 28(17):4528–4532, 2008.

[15] C. Sherstan, J. MacGlashan, and P. M. Pilarski. Generalizing value estimation over timescal. In *FAIM Workshop on Prediction and Generative Modeling in Reinforcement Learning*, 2018.

[16] P. D. Sozou. On hyperbolic discounting and uncertain hazard rates. *Proceedings of the Royal Society of London B: Biological Sciences*, 265(1409):2015–2020, 1998.

[17] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[18] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. 2018.

[19] S. C. Tanaka, K. Doya, G. Okada, K. Ueda, Y. Okamoto, and S. Yamawaki.

[20] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

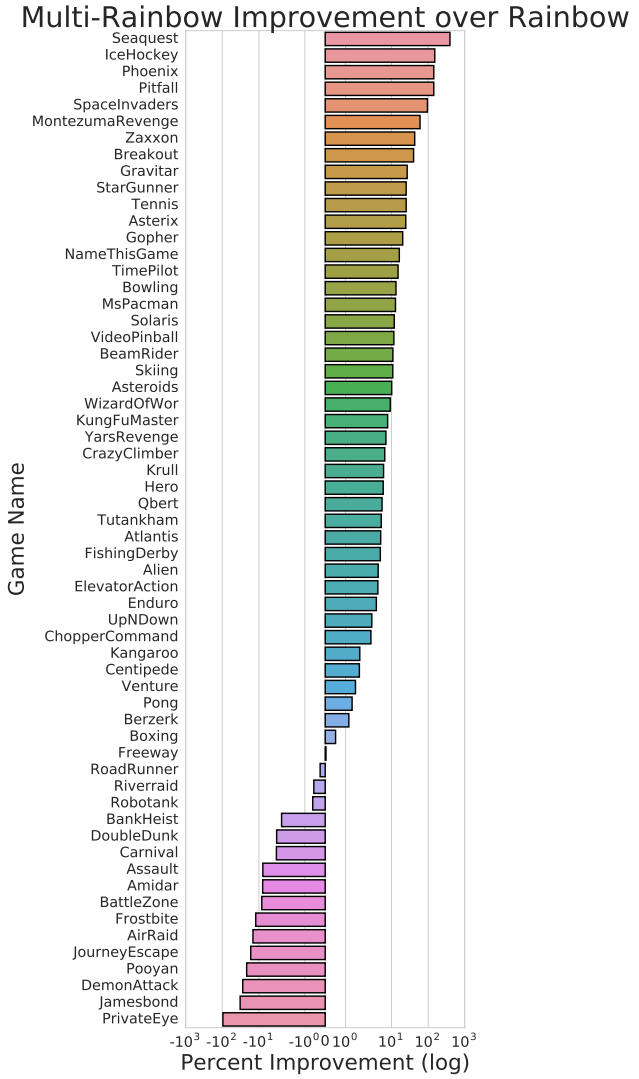


Figure 2: Performance improvement over Rainbow using the multi-horizon auxiliary task in Atari Learning Environment (3 seeds each).

Compositional subgoal representations

Carlos G. Correa
Princeton Neuroscience Institute
Princeton University
Princeton, NJ 08544
cgcorrea@princeton.edu

Fred Callaway
Department of Psychology
Princeton University
Princeton, NJ 08544
fredcallaway@princeton.edu

Mark K. Ho
Department of Psychology
Princeton University
Princeton, NJ 08544
mho@princeton.edu

Thomas L. Griffiths
Departments of Psychology and Computer Science
Princeton University
Princeton, NJ 08544
tomg@princeton.edu

Abstract

When faced with a complex problem, people naturally break it up into several simpler problems. This hierarchical decomposition of an ultimate goal into sub-goals facilitates planning by reducing the number of factors that must be considered at one time. However, it can also lead to suboptimal decision-making, obscuring opportunities to make progress towards multiple subgoals with a single action. Is it possible to take advantage of the hierarchical structure of problems without sacrificing opportunities to kill two birds with one stone? We propose that people are able to do this by representing and pursuing multiple subgoals at once. We present a formal model of planning with compositional goals, and show that it explains human behavior better than the standard "one-at-a-time" subgoal model as well as non-hierarchical limited-depth search models. Our results suggest that people are capable of representing and pursuing multiple subgoals at once; however, there are limitations on how many subgoals one can pursue concurrently. We find that these limitations vary by individual.

Keywords: planning; problem solving; hierarchy; goals

Acknowledgements

1 Introduction

It's Tuesday afternoon and you have a list of errands to run before you can return home to watch last night's episode of *The Bachelor*. You need to mail a letter, pick up broccoli for tonight's stir fry, and drop off a book at the library. You are eager to get home to see whether Hannah B. received one of the coveted roses, and thus want to accomplish these tasks as expediently as possible. There are two library locations, four grocery stores, and who knows how many mail boxes in your town; how do you decide which location of each to visit, and in what order? Unfortunately, you have been presented with the generalized traveling salesman problem, which is known to be NP-hard (that is, potentially very difficult to solve). You might simplify the problem by focusing on only one errand at a time, completing it as quickly as possible from wherever the last errand left you. But by taking this strategy you might miss opportunities to save time overall–e.g., by going to the library location that is further from your house, but near the grocery store.

The ability of people to plan and problem solve can be formulated in terms of search over an internal representation of the problem space (Newell & Simon, 1972). However, the combinatorial explosion of possible action sequences makes exhaustive search intractable, necessitating approximation strategies (Huys et al., 2015). One particularly valuable strategy for directing search is *subgoaling* (Donnarumma, Maisto, & Pezzulo, 2016). By focusing on a single subgoal at a time, one can ignore irrelevant actions and features of the environment, effectively reducing the dimensionality of the problem space (Dietterich, 2000). However, this reduction comes at a cost. Standard models of goal-setting and hierarchical planning assume that subgoals are *context-free* in the sense that they are pursued independently, without consideration of the ultimate goal or future subgoals. As a result, opportunities to make progress towards multiple subgoals at once will be missed. Nevertheless, people often *do* identify these opportunities. This poses a problem for the standard one-at-a-time model of subgoal pursuit.

How might people reap the fruits of hierarchical decomposition without being hamstrung by the artificial boundaries it imposes on problem solving? One possibility is that, contrary to the standard model, people are not limited to pursuing one goal at a time. That is, perhaps they choose a *subset* of goals, and construct a plan that is optimal with respect to that subset. They might ask themselves, for example, "What is the fastest way to get to both a grocery store and a library?", leaving future goals such as dropping off a letter for future consideration. We formalize this theory in a hierarchical planning framework. At the abstract level, the agent dynamically constructs composite goals (or *multigoals*) from a set of primitive goals. Then, at the concrete level, the agent searches for a plan that achieves the composite goal, disregarding future goals.

Here, we first discuss previous attempts to capture how people use subgoals to decompose problems. We then present a formalization of the multigoal model as a type of hierarchical planning. To test this account, we assess how well it explains human behavior on a classic problem solving task (Shallice, Broadbent, & Weiskrantz, 1982) and use a formal model comparison to show how it acccounts for participant behavior better than alternative models.

2 Background

Much previous work in psychology studies how people use representations to make problem solving tractable. We focus on two non-exclusive classes of strategies that have been studied: *tree search* and *hierarchical decomposition*. Tree search strategies take inspiration from algorithms based on search through a decision tree, in which possible plans are simulated forward and evaluated (Newell & Simon, 1972). For example, rather than consider plans that are arbitrarily long, people can engage in *depth-limited search* that focuses resources on plans that can be taken within a certain time horizon (Keramati, Smittenaar, Dolan, & Dayan, 2016; Krusche, Schulz, Guez, & Speekenbrink, 2018; MacGregor, Ormerod, & Chronicle, 2001). Alternatively, they may rely on local signals of progress to avoid less promising plans without explicitly considering them (Huys et al., 2012).

However, approaches such as depth-limited search and heuristics are limited by their locality—what if the goal is in the distant future, or even just out of reach of an arbitrary depth limit? What if the local cues are uninformative or simply do not exist? This, in part, motivates complementary approaches based on *hierarchical decomposition*, in which a problem is holistically broken into sub-problems, each of which is then solved independently (Botvinick, 2012; Sacerdoti, 1974). Such an approach can make decision-making easier by reducing the resources needed to solve a given sub-problem at a particular time (Maisto, Donnarumma, & Pezzulo, 2015; Van Dijk, Polani, & Nehaniv, 2009). But this raises new problems: What is a good way to break down a task? And how exactly should the identified hierarchy guide the choice of actions?

3 A Model of planning with multigoals

Multigoals extend standard goal-based hierarchical planning by allowing new subgoals to be constructed from primitive subgoals on the fly. Formally, we define a multigoal, $\mathcal{G}$, as a set of primitive subgoals. Following the options framework (Sutton, Precup, & Singh, 1999), a subgoal g is defined by $\beta_g : \mathcal{S} \rightarrow \{0,1\}$, which indicates whether a state satisfies the subgoal. A multigoal is satisfied when all component subgoals are satisfied: $\beta_{\mathcal{G}}(s) = \prod_{g \in \mathcal{G}} \beta_g(s)$. Here, we focus on the simplest case of a two-level hierarchy. At the abstract level the agent chooses a multigoal based on the current state and ultimate goal. At the concrete level, the agent attempts to find a sequence of actions that satisfies the current multigoal.

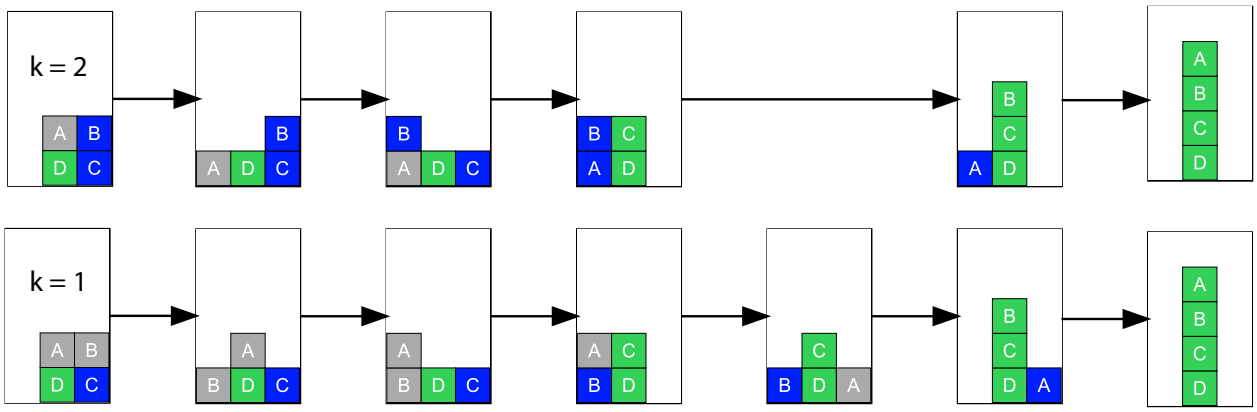


Figure 1: Planning with multigoals of different size k can lead to different action sequences. In this example, planning with $k = 2$ ensures that A is moved first which avoids extraneous actions at future states. Green blocks represent subgoals that have been satisfied and blue blocks represent the subgoals in the current multigoal, used to plan the next action.

To focus on how multigoals are composed from subgoals, here, we assume that possible subgoals and their appropriate ordering are given. Formally, the agent receives an ordered set of subgoals $\{g_1, g_2, \ldots g_n\}$ such that completing all n subgoals in order amounts to solving the ultimate goal. The abstract-level phase of planning is thus reduced to selecting k subgoals to pursue. Given k and the multigoal, $\mathcal{G}(s;k)$ is simply the set of the next k incomplete subgoals, as illustrated in Figure 1. For example, if g_1 has been completed ($\beta_{g_1}(s) = 1$) and $k = 2$, we would have $\mathcal{G}(s;k = 2) = \{g_2, g_3\}$. Admittedly, our assumption that possible subgoals and their ordering are given eliminates a considerable amount of interesting complexity. However, it allows us to study multigoals without tackling the full problem of multigoal selection. Relaxing this assumption is an important direction for future work.

Given a multigoal $\mathcal{G}$, concrete-level planning uses tree search to find a lowest-cost path π that satisfy all subgoals in $\mathcal{G}$. This search process may be depth-limited. If no path of length d or less can be found that satisfies all the subgoals, the model chooses a path that satisfies the largest possible number of subgoals. Thus, the action sequence is chosen by

$$\arg\max_{\pi} \sum_{g \in \mathcal{G}} \beta_g(s_\pi) - \lambda|\pi| \quad \text{s.t. } |\pi| \leq d, \tag{1}$$

where s_π is the final state on the path defined by the action sequence π and λ is a very small positive constant, with the effect that π is chosen to maximize the first term (number of subgoals completed), using the second term (path length) as a tie-breaker.

There are several special cases of the model worth highlighting. When $k = 1$, we recover a standard hierarchical planning algorithm in which only one subgoal is pursued at a time. When $k = \infty$, we recover a standard depth-limited search model with a heuristic cost function defined by the number of incomplete subgoals. When $d = \infty$, the model optimally achieves each multigoal, but may still perform suboptimally because future subgoals are not considered. And when $d = k = \infty$, the model always makes optimal choices.

4 Experiment: Measuring human multigoaling

To test the multigoal model, we conducted a Tower of London (ToL) experiment (Shallice et al., 1982) where an initial configuration of stacked items must be rearranged to match a goal configuration by moving one item at a time. Our variant of ToL uses blocks instead of disks and removes the restrictions on stack height. We additionally mark the blocks with letters and use a single standard goal position to reduce working memory requirements. This results in an unambiguous subgoal ordering (Kaller, Rahm, Kstering, & Unterrainer, 2011): One must put blocks in their place in reverse alphabetical order.

4.1 Methods

Each trial, participants were shown two configurations of stacked blocks marked with letters and asked to rearrange the blocks in one stack to match the blocks in the goal configuration (Figure 2a). Only the top block of a stack can be moved, and the board is limited to three columns. In all trials, the goal configuration is a single stack arranged in alphabetical order in the middle column.

Participants completed 3-, 4-, and 5-block tutorial trials and then completed sixteen 6-block test trials. Trials did not have a time limit. Only test trials were analyzed. Problems were selected to maximize the difference in likelihood of the fixed k multigoal model and the depth-limited model on data simulated from multigoal agents with $k \in \{1, 2, 3, 4\}$. We recruited 41 participants from Amazon Mechanical Turk. Each participant received $2 for completing the task, which took an average of 9 minutes.

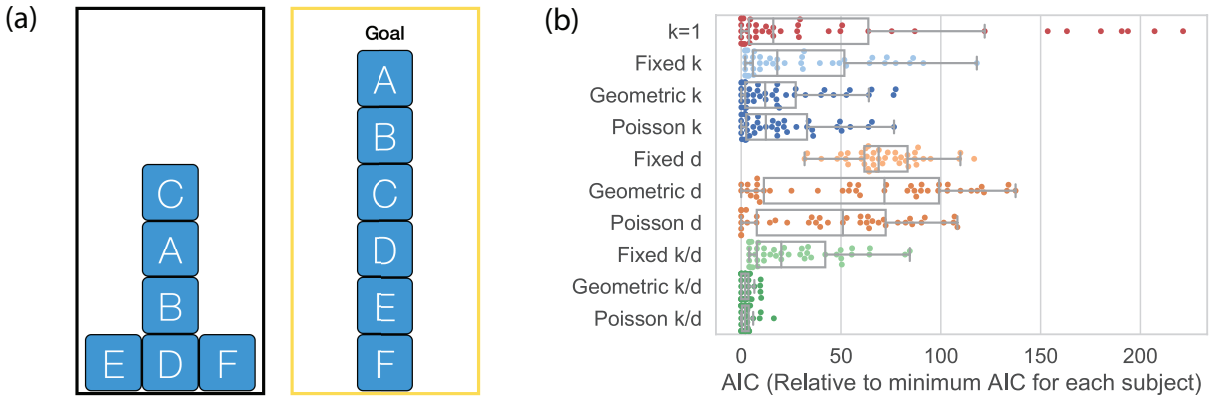


Figure 2: Experiment. (a) Experimental interface. (b) Relative model performance. Each point is one model fit to one participant. For ease of comparison, we normalize the AIC (Akaike information criterion) values for each participant by the AIC of their best fitting model. The same additive transformation is applied to the AIC values for each model, and thus the mean differences between models are not affected.

4.2 Model

According to the multigoal model, people select multiple subgoals to pursue concurrently, attempting to create a plan that satisfies all k subgoals in the fewest steps. A subgoal in the experimental task corresponds to putting a block in its correct final position with all prior subgoals completed (this ensures that subgoals will only be completed in order). Thus, we have an ordered set of subgoals $\{g_F, g_E, g_D, g_C, g_B, g_A\}$ such that completing them in order necessarily solves the full problem.

We derive model likelihoods as follows. Given a state and a value for k, the model selects a multigoal $\mathcal{G}(s;k)$ which is the set of the next k uncompleted subgoals. Then, given this multigoal and a value for d, we find a set of optimal paths given by Equation 1. The model uniformly selects among optimal actions or selects an action randomly with probability ε.

To test the multigoal model's prediction that people can pursue more than one goal at a time, we employ formal model comparison. The comparisons of greatest interest are special cases of the full model. A standard depth-limited search model sets $k = \infty$ and has d (the search depth) as a free parameter. A standard hierarchical planning or subgoaling model sets $k = 1$, allowing for only one subgoal to be pursued at a time, and also has d as a free parameter. In the full model, both k and d are free.

We consider two hypotheses for how k and d are chosen when they are free parameters. The first hypothesis is that the parameter is fixed for each individual. For example, one person might always consider two subgoals at a time ($k = 2$) and search fourteen steps into the future ($d = 14$). The second hypothesis is that k and d vary from decision to decision, but follow some distribution that is particular to each individual. For example, one person might typically consider one subgoal at a time but occasionally pursue two (or rarely, three) subgoals at a time. We consider two possible distributions for both k and d: the Geometric distribution, which has a strong positive skew, and the Poisson distribution, which can distribute mass more evenly around a mean. Because these distributions both have a single parameter, they do not increase the number of parameters in the full model. To compute the likelihood, we integrate out the latent k and d parameters. We fit all models by maximum likelihood estimation. For discrete parameters we used grid search, considering all possible values of $k \in \{1 \ldots 6\}$ and values of $d \in \{1 \ldots 15\}$. We chose 15 as this was the farthest a participant ever was from a goal state. For continuous parameters, we used L-BFGS.

4.3 Results

Models were fit separately for each participant. Results are summarized in Figure 2. We highlight three primary results below.

First, all models that include multigoals explain the data significantly better than the $k = 1$ model. This suggests that people are not limited to pursuing one subgoal at a time. That being said, for some participants, the best fitting p_k parameter in the Geometric k/d model was close to one, resulting in single subgoal planning in most cases. In particular p_k was greater than 0.95 for 17 out of 41 participants, suggesting that these participants considered more than one subgoal at most 5% of the time.

Second, all models with free k parameters performed better than the models with free d parameters. This suggests that limits on composite subgoal representations provide a better explanation for participants' behavior than limits on the depth of search. However, models that include both kinds of limits perform best. This suggests that human problem solving may be jointly constrained by subgoal representations and search depth, and that these two constraints each provide unique contributions to explaining behavior.

Third, models with distributions over model parameters k and d always performed better than the corresponding models with fixed parameters. This suggests that people consider different numbers of subgoals and search to varying depths at different points. This effect could be due to changing problem structure: The configuration of blocks at any moment may determine, for example, the

number of subgoals a person is capable of considering. An alternative (although not mutually exclusive) explanation is that our participants adaptively adjusted the complexity of their multigoals, choosing how much effort to put into the problem based on an estimate of how much doing so will improve their performance (Lieder & Griffiths, 2017; Shenhav et al., 2017).

5 Discussion

People often solve complex problems by breaking them down into simpler problems. The importance of subgoals has been long recognized, and they play a critical role in the most successful models of human problem solving (Anderson, 2013; Laird, Newell, & Rosenbloom, 1987; Newell & Simon, 1972). However, these models are limited by the assumption that only one subgoal can be pursued at a time. To address this limitation, we have extended the classic subgoaling model with the notion of a multigoal, a composite subgoal that is constructed on the fly and pursued concurrently. Through a formal model comparison, we showed that the multigoal model explains data better than a standard one-at-a-time subgoal model or a model that ignores subgoal structure altogether.

We have presented a theory of how people might represent and pursue multiple concurrent subgoals, and we have provided preliminary evidence that they do this. Future work will need to determine in greater detail how people select which subgoals to pursue at any moment. Thus, in the context of the experiment, we treat k as a latent variable to be inferred, not predicted. A more complete model would explain why people choose the k they do, perhaps as a function of problem complexity or stakes. Such a theory could be constructed within the framework of resource-rational analysis (Griffiths, Lieder, & Goodman, 2014), predicting that people choose a k to optimize a trade-off between the efficiency of the problem solution and the cost of representing and pursuing more complex multigoals. An important open question is how to quantify the cost of a multigoal, or even a standard subgoal.

Another major direction for future work is to remove the simplifying assumption of subgoal-ordering. This assumption allows us to reduce the problem of composing multigoals from a set of subgoals to the problem of selecting a single integer k, greatly simplifying model specification and inference. However, many of the problems people are faced with do not have such an assumed ordering. To return to the initial motivating example, you do not know a priori whether you should go to a grocery store or library first. Unfortunately, choosing a multigoal from an unordered set of subgoals is a daunting task. With n basic subgoals, there are $\binom{n}{k}$ possible multigoals of size k that you could construct. But fortunately, as Colton demonstrates when he hands out roses at the end of each episode, people are entirely of capable of choosing k out of n options.

References

Anderson, J. R. (2013). *The architecture of cognition*. Psychology Press.

Botvinick, M. M. (2012). Hierarchical reinforcement learning and decision making. *Current opinion in neurobiology*, *22*(6), 956–962.

Dietterich, T. G. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303. doi: 10.1613/jair.639

Donnarumma, F., Maisto, D., & Pezzulo, G. (2016). Problem Solving as Probabilistic Inference with Subgoaling: Explaining Human Successes and Pitfalls in the Tower of Hanoi. *PLoS Computational Biology*, *12*(4), 1–30. doi: 10.1371/journal.pcbi.1004864

Griffiths, T. L., Lieder, F., & Goodman, N. D. (2014). Rational use of cognitive resources: Levels of analysis between the computational and the algorithmim. *Topics in Cognitive Science*, *7*(2), 217–229. doi: 10.1111/tops.12142

Huys, Q. J. M., Eshel, N., O'Nions, E., Sheridan, L., Dayan, P., & Roiser, J. P. (2012). Bonsai trees in your head: how the Pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS Comput Biol*, *8*(3), e1002410.

Huys, Q. J. M., Lally, N., Faulkner, P., Eshel, N., Seifritz, E., Gershman, S. J., ... Roiser, J. P. (2015). Interplay of approximate planning strategies. *Proceedings of the National Academy of Sciences of the United States of America*, *112*(10), 3098–103. doi: 10.1073/pnas.1414219112

Kaller, C. P., Rahm, B., Kstering, L., & Unterrainer, J. M. (2011). Reviewing the impact of problem structure on planning: A software tool for analyzing tower tasks. *Behavioural Brain Research*, *216*(1), 1 - 8.

Keramati, M., Smittenaar, P., Dolan, R. J., & Dayan, P. (2016, nov). Adaptive integration of habits into depth-limited planning defines a habitual-goaldirected spectrum. *Proceedings of the National Academy of Sciences*, *113*(45), 12868–12873. doi: 10.1073/pnas.1609094113

Krusche, M. J. F., Schulz, E., Guez, A., & Speekenbrink, M. (2018). Adaptive planning in human search. *bioRxiv*, 0–5. doi: 10.1101/268938

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial intelligence*, *33*(1), 1–64.

Lieder, F., & Griffiths, T. L. (2017). Strategy selection as rational metareasoning. *Psychological review*, *124*(6), 762.

MacGregor, J. N., Ormerod, T. C., & Chronicle, E. P. (2001). Information processing and insight: a process model of performance on the nine-dot and related problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *27*(1), 176.

Maisto, D., Donnarumma, F., & Pezzulo, G. (2015, feb). Divide et impera: subgoaling reduces the complexity of probabilistic inference and problem solving. *Journal of The Royal Society Interface*, *12*(104), 20141335–20141335. doi: 10.1098/rsif.2014.1335

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, *5*(2), 115–135.

Shallice, T., Broadbent, D. E., & Weiskrantz, L. (1982). Specific impairments of planning. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, *298*(1089), 199-209. doi: 10.1098/rstb.1982.0082

Shenhav, A., Musslick, S., Lieder, F., Kool, W., Griffiths, T. L., Cohen, J. D., & Botvinick, M. M. (2017). Toward a Rational and Mechanistic Account of Mental Effort. (March), 99–124.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, *112*(1-2), 181–211.

Van Dijk, S. G., Polani, D., & Nehaniv, C. L. (2009). Hierarchical behaviours: getting the most bang for your bit. In *European conference on artificial life* (pp. 342–349).

Skynet: A Top Deep RL Agent in the Inaugural Pommerman Team Competition

Chao Gao,[*] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor
Borealis AI
Edmonton, Alberta, Canada
{pablo.hernandez, bilal.kartal, matthew.taylor}@borealisai.com

Abstract

The Pommerman Team Environment is a recently proposed benchmark which involves a multi-agent domain with challenges such as partial observability, decentralized execution (without communication), and very sparse and delayed rewards. The inaugural Pommerman Team Competition held at NeurIPS 2018 hosted 25 participants who submitted a team of 2 agents. Our submission `nn_team_skynet955_skynet955` won 2nd place of the "learning agents" category. Our team is composed of 2 neural networks trained with state of the art deep reinforcement learning algorithms and makes use of concepts like reward shaping, curriculum learning, and an automatic reasoning module for action pruning. Here, we describe these elements and additionally we present a collection of open-sourced agents that can be used for training and testing in the Pommerman environment. Relevant code available at: `https://github.com/BorealisAI/pommerman-baseline`

Keywords: Deep Reinforcement Learning; Multi-Agent Deep Reinforcement Learning; Pommerman

Acknowledgements

[*]cgao3@ualberta.ca Department of Computing Science, University of Alberta. Work performed as intern at Borealis AI

1 Introduction

The Pommerman environment for benchmarking Multi-Agent Learning is based on the classic console game **Bomberman**. The team competition involves 4 bomber agents initially placed at the four corners of an 11×11 board. Each two diagonal agents form a team. At every step, each agent issues an action simultaneously from 6 discrete candidate moves: moving `left`, `right`, `up`, `down`, placing a `bomb`, or `stop`. The `bomb` action is legal as long as the agent's `ammo` is greater than 0, and any illegal action is superseded with `stop` by the environment. Each cell on the board can either be a `passage`, a `rigid` wall, or `wood`. Only `passage` is passable for the agent. `Wood` would be destroyed if it is covered by an exploding bomb's flame while a `rigid` location is unaffected. Also, when `wood` is destroyed a `powerup` might appear (according to some probability) at the same location. There are 3 types of powerups: `ExtraBomb`, `EnableKick`, and `ExtraBlast` for which the agent respectively, increases its ammo by 1, acquires the ability to kick bombs, and rises its bombs' blast strength by 1.

The game starts with a procedurally generated random map and each agent initially has an `ammo` of 1 and `blast strength` of 2. Whenever an agent places a bomb, it explodes after 10 time steps, producing flames that have a lifetime of 2 time steps and a radius of the bomb placing agent's `blast strength`. The game ties when both teams have at least one agent alive after 800 steps. The team environment is also partially observable, meaning each agent can only see the local board with a radius of 4 cells, see Figure 1(a).

Pommerman is a challenging benchmark for multi-agent learning and model-free reinforcement learning, due to the following characteristics:

Sparse and deceptive rewards: the former refers to the fact that the only non-zero reward is obtained at the end of an episode. The latter refers to the fact that quite often a winning reward is due to the opponents' involuntary suicide, which makes reinforcing an agent's action based on such a reward *deceptive*. Note that suicide happens frequently during learning since an agent has to place bombs to explode wood to move around on the board, while due to terrain constraints, in some cases, performing non-suicidal bomb placement requires complicated, long-term, and accurate planing.

Delayed action effects: the only way to make a change to the environment (e.g., bomb wood or kill an agent) is by means of bomb placement, but the effect of such an action is only observed when the bomb's timer decreases to 0; more complications are added when a placed bomb is kicked to another position by some other agent.

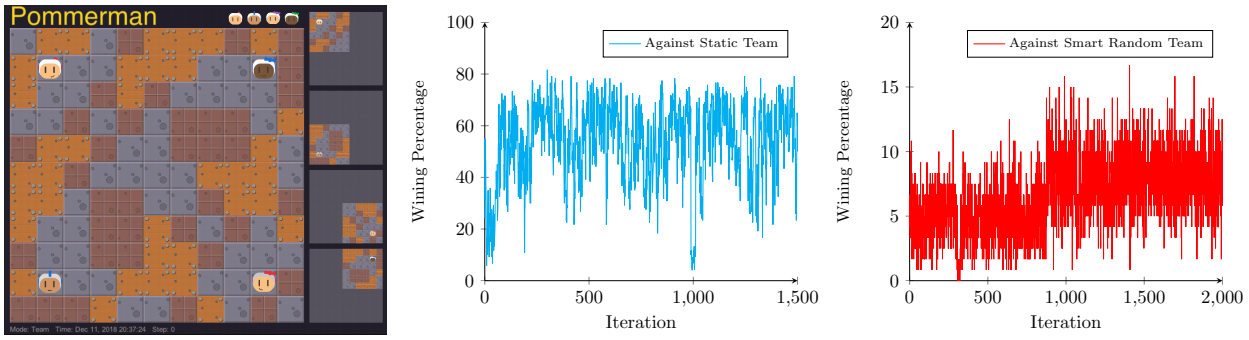
Imperfect information: an agent can only see its nearby areas. This makes the effect of certain actions, such as bomb kicking, unpredictable. Indeed, even detecting which agent placed the exploding bomb is intractable in general because of the hidden information of the board.

Uninformative multiagent credit assignment: In the team environment, the same episodic reward is given to two members of the team. It may not be clear how to assign credit to individual agents. For example, consider an episode where an agent eliminates an opponent but then commits suicide, and its teammate eliminates the remaining opponent. Under this scenario, both team members get a positive reward from the environment, but this could reinforce the suicidal behaviour of the first agent. Similarly, one agent could eliminate both opponents whereas its teammate just camps; both agents would get positive rewards, reinforcing a *lazy* agent [7].

2 Skynet955

`nn_team_skynet955_skynet955` is a team composed of two identical neural networks, where `skynet955` is trained after equipping the neural net agent with an "ActionFilter" module for 955 iterations. The philosophy is to instill prior knowledge to the agent by telling the agent *what not to do* and let the agent discover *what to do* by trial-and-error. The benefit is twofold: 1) the learning problem is simplified since suicidal actions are removed and bomb placing becomes safe; and 2) superficial skills such as not moving into flames and evading bombs in simple cases are handled. Below we describe the main components of our team: the ActionFilter and the reinforcement learning aspect.

ActionFilter We designed the filter to speed up learning so that agents can focus on higher level strategies. The filter thus serves as a safety check to provide more efficient exploration. The ActionFilter is implemented by rules shown in Table 1. Note that for the "avoiding suicide" rules, `skynet955` implemented a simple version of them (e.g., a moving bomb was simply treated as static); a full implementation would arguably make the agent stronger. It is worth mentioning that the above ActionFilter is extremely fast. In our experiments, together with neural net evaluation, each action takes ≈ 3 ms on a GTX 1060 GPU on average, while the time limit in the competition is 100 ms per move. Also, we note that another natural approach for "bomb placement" pruning is conducting a lookahead search using "avoding suicide" rules; this is perhaps better than the crude rules described above. We think this ActionFilter will be useful for learning agents within the Pommerman domain, since it can constrain the action space during learning without significantly affecting the overall team strategy. Ideally, one can even start model-free learning with the filter, and once an agent acquires some basic



(a) (b)

Figure 1: (a) An example team environment Pommerman game. On the right is shown the partial observation (a limited area around) for each agent. (b) Winning percentage of the learning team against a team of `Static` opponents and a team of `Smart` random opponents.

Table 1: ActionFilter rules

Avoiding Suicide	Not going to positions that are flames on the next step. Not going to *doomed* positions, i.e., positions where if the agent were to go there the agent would have no way to escape. For any bomb, doomed positions can be computed by referring to its `blast strength`, `blast range`, and `life`, together with the local terrain.
Bomb Placement	Not place bombs when teammate is close, i.e., when their Manhattan distance is less than their combined blast strength. Not place bombs when the agent's position is covered by the blast of any previously placed bomb.

skills, it can be unplugged from the neural network so that strategy bias by filter is completely removed. For this reason, we have open-sourced an implementation of the ActionFilter.[1]

Reinforcement Learning As depicted in Figure 2, the architecture contains 4 convolution layers, followed by policy and value heads. The input contains 14 features planes, each of shape 11×11, similar to [9]. It then convolves using 4 layers of convolution, each has 64 3×3 kernels; the result thus has shape $11\times11\times64$. Then, each head convolves using 2 1×1 kernels. Finally, the output is squashed into action probability distribution and value estimation, respectively. Such a two-head architecture is a natural choice for Action-Critic algorithms, as it is generally believed that forcing policy and value learning to use shared weights could reduce over-fitting [10].

Instead of using an LSTM to track the history observations, we use a "retrospective board" to remember the most recent value at each cell of the board, i.e., for cells outside of an agent's view, in the "retrospective board" its value is filled with what was present when the agent saw that cell the last time. The input feature has 14 planes in total, where the first 10 are extracted from the agent's current observation, the rest 4 are from "retrospective board." We initially performed experiments with an LSTM to track all previous observations; however, due to computational overhead on top of the already prolonged training in Pommerman domain, we decided to replace it with the "retrospective board" idea, which yielded similar performance but was significantly faster.

[1] `https://github.com/BorealisAI/pommerman-baseline`

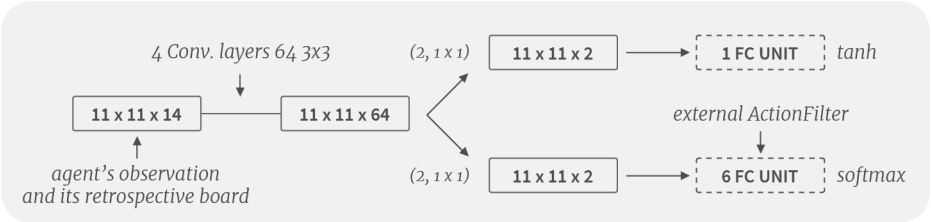


Figure 2: Architecture used for the `skynet955` agents

Table 2: Reward Shaping for `skynet955` agents

Going to a cell not in a 121-length FIFO queue gets 0.001.	At the end of a game, dead agent in the winning team gets 0.5.
Picking up `kick` gets 0.02.	For draw games, all agents receive 0.0.
Picking up `ammo` gets 0.01.	On one enemy's death gets 0.5.
Picking up `blast strength` gets 0.01.	On a teammate's death gets -0.5.

The neural net is trained by *PPO* [10], minimizing the following objective:

$$o(\theta;\mathcal{D}) = \sum_{(s_t,a_t,R_t)\in\mathcal{D}} \left[-clip(\frac{\pi_\theta(a_t|s_t)}{\pi_\theta^{old}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)A(s_t,a_t) + \frac{\alpha}{2}\cdot\max\left[(v_\theta(s_t)-R_t)^2, (v_\theta^{old}(s_t)+clip(v_\theta(s_t)-v_\theta^{old}(s_t),-\epsilon,\epsilon)-R_t)^2\right]\right], \tag{1}$$

where θ is the neural net, $\mathcal{D}$ is sampled by π_θ^{old}, and ϵ is a tuning parameter. Refer to OpenAI baseline for details [10].

Curriculum Learning Training is conducted by letting two identical neural net players compete against a set of curriculum [1] opponents: (i) `Static` opponent teams, where opponents do not move or place bombs. Competing against a team of `Static` opponents teaches our agents to get closer to opponents, place a bomb, and move away to a safe zone. The trained neural net is then used against the second opponent in the curriculum. (ii) `SmartRandomNoBomb`: players that do not place bombs. *Smart* random means it has the ActionFilter as described earlier and the action taken is random (except that bomb placing is disallowed). The reason we let `SmartRandomNoBomb` not place bombs is that the neural net can focus on learning true "killing" skills, not a skill that solely relies on the opponent's strategy flaw (e.g., the provided baseline `SimpleAgent` has a significant flaw where the competitor can diagonally block and make `SimpleAgent` be killed by its own bomb). This avoids the "false positive" reward signal caused by opponent's involuntary suicide. Competing against a team of `SmartRandomNoBomb` helps our agents to learn better battling skills such as using the topological map to corner the opponents and pursuing opponents.

Reward shaping: To cope with the sparse reward problem, a dense reward function is added during the learning, see Table 2. It should be noted the above hand designed reward function is still noisy in the sense that an agent's contribution was not clearly separated.

Results: Figure 1(b) shows the learning curves against `Static` and `SmartRandomNoBomb` teams. In our training, each iteration contains 120 games, produced in parallel by 12 actor workers. The curves show that, against `Static` agents, the neural net achieved wining percentage around 70%, while against `SmartRandomNoBomb`, it never reached 20%. We note that because the opponents do not place bombs, the rest of the games are almost all draws. The learning seems to be slow, in part because playing against `SmartRandomNoBomb`, a large number of games were ended with draws, which gives reward signal 0 in our training.

In the competition, our team was composed of two identical neural net models, at each step, for each of our agent, each action typically costs one to several milliseconds, while the time limit is 100ms per move. The submitted agent `skynet955` was the neural net model at iteration 955 obtained in training against `SmartRandomNoBomb` team, as by the time of submission only around 1000 iterations were finished. Throughout our training and testing, only the `V0` environment (which has no wall collapsing) was used. By contrast, in the competition, the `V1` (which has wall collapsing, meaning at certain time step, the boarder passages will suddenly change to rigid walls, and any agent that happens to be in any of the corresponding cells dies) was used.

An open-sourced collection of agents for Pommerman Against `SimpleAgent`, it is not difficult to train a non-placing-bomb neural net agent that wins by diagonally blocking and forcing `SimpleAgent` to get stuck on its self-placed bomb.[2] This strategy flaw of `SimpleAgent` stems from its hand-crafted heuristic strategy for enemy engagement and bomb placement. Learning agents exploit this flaw and the learned policy does not generalize against other opponents types. Therefore, we propose and open source different agents to be used for training, see Table 3.

`StaticAgent` is an extremely simple agent that always executes the stop action, the advantage of using this agents is that rewards are noise-free. `SmartRandomNoBomb` agent is a very challenging opponent. It moves randomly among the filtered actions, therefore learning against it provides better generalization. It does not place bombs, and thus never commits suicide. In contrast to `SimpleAgent` and `CautiousAgent`, it does not use Dijkstra, and it takes less time to act.

`CautiousAgent` is based on a modification of `SimpleAgent`, the main idea is to only let this agent place a bomb when it is certain to kill an opponent. However, with this adaptation, the "weak" opponent `SimpleAgent` is instantly turned into a "strong" player. Killing this agent requires quite advanced skills. When learning against this opponent, a large number

[2] `https://youtu.be/3yUhI46Xx8o`

Table 3: Description of different agents. Our open-sourced agents: `StaticAgent`, `SmartRandom`, `SmartRandomNoBomb` and `CautiousAgent`, will be helpful to baseline against and to train against. The bottom 3 agents are relatively stronger than the provided baseline, less prone to exploitation due to higher level of stochasticity, and fast decision makers to be used during training.

`SimpleAgent` (Provided by Pommerman)	A heuristic agent that uses Dijkstra and rules for navigation, tune-up collection, and simple attacks.
`StaticAgent`	A boring agent that always executes the stop action. Helpful for learning agents as rewards are noise-free.
`SmartRandom` (Random agent + ActionFilter)	Takes random actions from the filtered action space. Stochastic but careful actions render it a competitive opponent for RL agents.
`SmartRandomNoBomb` (Random agent + ActionFilter + No Bombing)	Similar to `SmartRandom` but force the agent not place bomb at all.
`CautiousAgent` (Modified SimpleAgent)	The modification enables the agent to place a bomb if and only if it guarantees a kill.
`Skynet955` (Neural Network agent)	Agent that is trained with PPO using reward shaping, ActionFilter, and opponent curriculum learning.

of games ended with draws, or the challenger is killed. For example, in our tests, our `Skynet` agent can easily achieve 70% wining percentage against `SimpleAgent` team, but only around 10% against `CautiousTeam`, if not excluding draws.

3 Conclusions and Future work

In spite of the good performance of our `skynet` agents, there are still many potential avenues for future research. For example, recent innovations, such as curiosity [2, 8], centralized learning with decentralized execution [5], or difference rewards [4] have shown to be able to produce good results in related domains, it remains to experimentally verify how effective they would be in the challenging domain of Pommerman.

One challenge of multi-agent learning in partial observable environments is the credit assignment for each individual agent's behavior [3, 6]. Whenever there is a success or failure, it is important to correctly identify who is to reward or blame. Unfortunately, the original environment provided by Pommerman does not provide any such information. However, if centralized training [5] can be used by revising the environment, information might be helpful in devising more accurate reward function, for example: identification of bombs' owners and bombs' kickers; in general for any relevant event occurred (wood destruction, enemy's death, etc.) identifying which agent is to reward or blame could be based on who is responsible for the corresponding exploding bomb. Lastly, an ensemble consists of multiple neural net models may also improve the playing performance in the competition setting.

References

[1] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[2] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[3] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer. Potential-based difference rewards for multiagent reinforcement learning. In *AAMAS*, pages 165–172, 2014.

[4] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual Multi-Agent Policy Gradients. In *32nd AAAI Conference on Artificial Intelligence*, 2017.

[5] J. N. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*, 2017.

[6] P. Hernandez-Leal, B. Kartal, and M. E. Taylor. Is multiagent deep reinforcement learning the answer or the question? A brief survey. *arXiv preprint arXiv:1810.05587*, 2018.

[7] L. Panait and S. Luke. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems*, 11(3), Nov. 2005.

[8] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

[9] C. Resnick, W. Eldridge, D. Ha, D. Britz, J. Foerster, J. Togelius, K. Cho, and J. Bruna. Pommerman: A multi-agent playground. *arXiv preprint arXiv:1809.07124*, 2018.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Autonomous Open-Ended Learning of Interdependent Tasks

Vieri Giuliano Santucci[1], **Emilio Cartoni**[1], **Bruno Castro da Silva**[2], **Gianluca Baldassarre**[1]
[1]Istituto di Scienze e Tecnologie della Cognizione (ISTC)
Consiglio Nazionale delle Ricerche (CNR)
Roma , Italy
{vieri.santucci,emilio.cartoni,gianluca.baldassarre}@istc.cnr.it
[2]Institute of Informatics
Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre, Brazil
bsilva@inf.ufrgs.br

Abstract

Autonomy is fundamental for artificial agents acting in complex real-world scenarios. The acquisition of many different skills is pivotal to foster versatile autonomous behaviour and thus a main objective for robotics and machine learning. Intrinsic motivations have proven to properly generate a task-agnostic signal to drive the autonomous acquisition of multiple policies in settings requiring the learning of multiple tasks. However, in real-world scenarios tasks may be interdependent so that some of them may constitute the precondition for learning other ones. Despite different strategies have been used to tackle the acquisition of interdependent/hierarchical tasks, fully autonomous open-ended learning in these scenarios is still an open question. Building on previous research within the framework of intrinsically-motivated open-ended learning, we propose an architecture for robot control that tackles this problem from the point of view of decision making, i.e. treating the selection of tasks as a Markov Decision Process where the system selects the policies to be trained in order to maximise its competence over all the tasks. The system is then tested with a humanoid robot solving interdependent multiple reaching tasks.

Keywords: Interdependent Tasks, Hierarchical Skill Learning, Intrinsic Motivations, Reinforcement Learning, Autonomous Robotics

Acknowledgements

This project has received funding from the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement no. 713010 (GOAL-Robots – Goal-based Open-ended Autonomous Learning Robots). This work was also partially supported by the Brazilian FAPERGS under grant no. 17/2551-000.

1 Introduction

Autonomous acquisition of many different skills is necessary to foster behavioural versatility in artificial agents and robots. While the learning of multiple skills *per se* can be addressed through different machine learning techniques sequentially assigning a series of N tasks to the agent, autonomy implies that the agent itself has the capacity to select on which task to focus at each moment and to shift between them in a smart way. Intrinsic motivations (IMs) have been used in the field of machine learning and developmental robotics [15, 2] to provide self-generated reinforcement signals driving exploration and skill learning [15, 18, 9]. Other works [3, 19, 7] implemented IMs as a motivational signal for the autonomous selection of tasks (often called "goals"): the learning progress in accomplishing the tasks is used as a transient reward to select goals in which the system is making the most learning progress [10, 17].

In real-world scenarios, tasks may require specific initial conditions to be performed or may be interdependent, so that to achieve a task the agent needs first to learn and accomplish other tasks. This latter case is of particular interest and although it has been studied under different headings, it is still an open question from an autonomous open-ended learning perspective. Hierarchical reinforcement learning [4] has been combined with IMs to allow for the autonomous formation of skills sequences. These methods often tackle only discrete state and actions domains [20]; or focus on the discovery of sub-goals on the basis of externally-given tasks [1] or under the assumption that sub-goals come as predefined rewards [14], thereby reducing the autonomy of the agent learning process. Imitation learning methods have also achieved important results in learning task hierarchies [8, 12, 13], also in association with IMs [5], but by definition they rely on external knowledge sources (e.g., an "instructor"), which limits the agent's autonomy.

We propose a reinforcement learning (RL) system for robot control that is capable of learning multiple interdependent tasks by treating the selection of tasks/goals as a Markov Decision Process (MDP) where the agent selects goals to maximise its overall competence.

2 Problem Analysis and Suggested Solution

From an RL perspective, learning of multiple goals can be treated as learning different policies π_g, each one associated with a different goal state $g \in G$. Such policies aim at maximising the return provided by a reward function R_g associated with goal g (see e.g. [6]). For each g the system thus learn a policy

$$\pi_g^*(a|s) = \underset{\pi}{\text{argmax}}\, R_g(\pi_g) \tag{1}$$

Since we are considering an open-ended learning scenario where no specific tasks are assigned to the robot, we assume that the system does not aim at maximising extrinsic rewards, but rather a competence function C over the distribution of goals G. Here, C is the sum of the agent's competence C_g at each goal g as made possible by a given candidate goal-selection policy Π_t. In other words, competence is a measure of the agent's ability to efficiently accomplish different goals by allocating its time among them using a given policy Π_t. Each goal can thus be associated with an MDP where the agent is tasked with learning to maximise the competence C_g for that goal rather than the goal's extrinsic reward R_g. If we consider a finite time horizon T, the robot needs to properly allocate its training time to the goals that guarantee the highest competence gain. To do so, the system may use the current derivative of the competence δC (w.r.t. time) as an intrinsic motivation signal to select the goal with the highest competence improvement at each time step t, where time here refers to one training step over a given task. The problem of task selection can thus be described as an N-armed bandit (possibly a *rotting bandit* due to the non-stationary transient nature of IMs) where the agent learns a policy Π to select goals that maximise the current competence improvement δC:

$$\Pi^* = \underset{\Pi}{\text{argmax}}\, \delta C(\Pi_t) \tag{2}$$

The efficacy of this approach has been demonstrated in different works within the intrinsically motivated open-ended learning framework [10, 11, 16]. If we constrain the feasibility of the goals to specific environmental conditions, goal selection becomes a *contextual bandit* problem where the robot has to learn to select goals depending on its current state $s \in S$. Equation 2 thus becomes:

$$\Pi^*(s_t) = \underset{\Pi}{\text{argmax}}\, \delta C(\Pi(s_t)) \tag{3}$$

where now the policy for selecting the goals to train needs to explicitly take into account the current state of the agent, which may include information such as which other goals have already been accomplished. By making this change to the objective, the system can bias the choice using the expected competence gain for each goal given different conditions. The evaluation of the competence improvement for each goal can be done via a state-base moving average of performance at achieving that goal given the current policy. If we now further assume a situation where goals are *interdependent*, so that a goal may be a precondition for other ones, we shift to a different kind of problem where the state of the environment depends on previously selected (and possibly achieved) goals. A sequence of contextual bandits where the context at time $t+1$ is determined by the "action" (here, goal selection) executed at time t, can be seen as an MDP over all the

goal-specific MDPs for which the robot is learning the policy (a "skill"). This is a setting that hierarchical skill learning methods have only scarcely addressed within a fully autonomous open-ended framework.

In this paper we propose that, given the structure of the problem, goal selection in the case of multiple interdependent tasks can be treated as an MDP and, consequently, can be addressed via RL algorithms that transfer *intrinsic-motivation* values between interrelated goals. In particular, in the following sections we show how a system implementing goal selection with a standard Q-learning algorithm is able to outperform systems that treat goal selection as a standard bandit or contextual bandit problem.

3 Experimental Setup and System Comparison

To test our hypothesis we compared different goal-selection systems in a robotic scenario with a simulated iCub robot (Fig. 1) that has to reach and "activate" 6 different spheres. We present results comparing three algorithms (discussed below) in two experimental scenarios:
1) *Environmental Dependency/Contextual Bandit Setting*: the activation of a sphere, by having the robot touch it, is dependent on some environmental variable. In this setting we assume a state feature (the "contextual feature") that is set to 1.0 with 50% probability at the beginning of each trial, and to 0.0 otherwise. The environment is composed of a total of six spheres that the agent needs to learn to activate: three can only be activated when the contextual feature is on, and the other three only when it is off.
2) *Multiple Interrelated Tasks/MDP Setting*: the "achievability" of a task (activation of a sphere) is now dependent on the activation status of the other spheres. In this scenario, the fact that the robot has previously achieved or not a goal (or set of goals) constitutes the precondition for the achievement of other goals, thus introducing interdependencies between the available tasks.

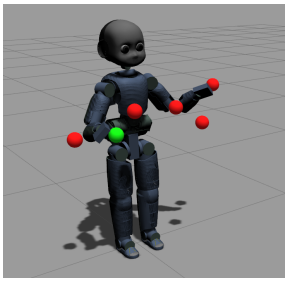


Figure 1: The simulated iCub robot in our experimental setup: when a sphere is touched (given its preconditions) it "lights up", changing its colour to green.

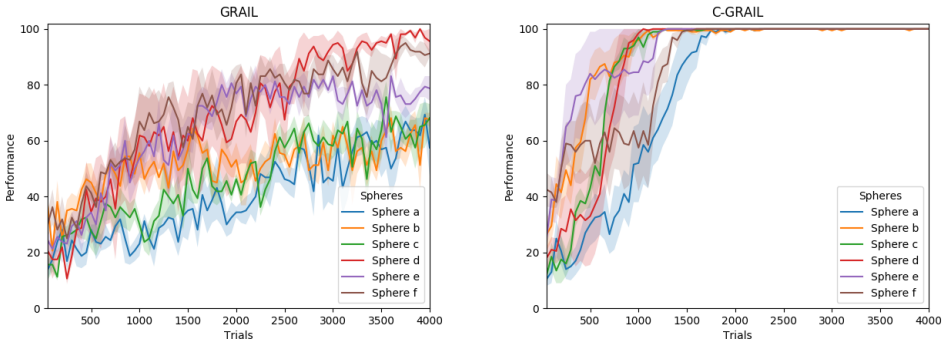


Figure 2: Performance of GRAIL and C-GRAIL in the first experiment

In this paper we compare three goal-selection systems that build upon the existing GRAIL architecture [19]. This architecture is generally composed of two components: a high-level component, called the "goal selector" (GS), which performs task selection on the basis of competence-based intrinsic motivations; and a low-level component composed of a set of low-level experts (one per task or goal); each expert is an actor-critic neural network implementing a candidate policy for achieving a goal. In the original version of GRAIL, the GS component receives no input from the environment

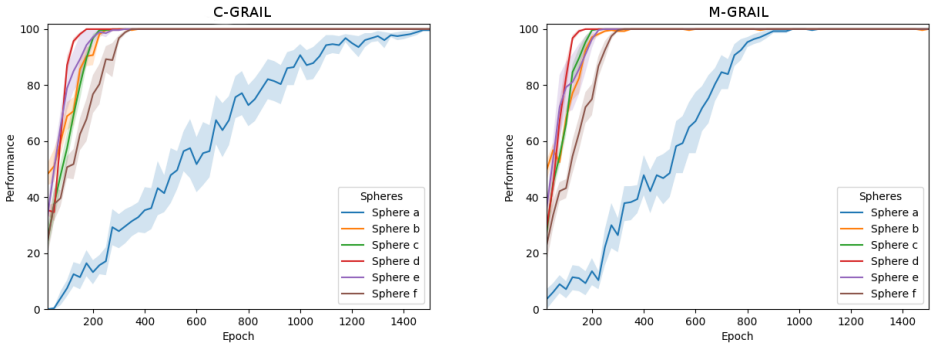


Figure 3: Performance of C-GRAIL and M-GRAIL over the 6 goals in the second experiment

and selects goals as in a standard bandit setting, where each arm/goal is evaluated on the basis of an exponential moving average (EMA) of the previously-acquired intrinsic rewards associated with achieving that goal. We now discuss two different versions of GRAIL that, by modifying the GS component, are able to cope with the added complexity of the scenarios described above. The first, called *Contextual*-GRAIL (C-GRAIL), provides as input to the GS the state of the environment, which can be composed of standard state features or also features describing the status of different goals (e.g features describing whether each sphere is activated). The GS then selects tasks to practice as in a contextual bandit where different EMAs are associated with different contexts. A second possible modification to GRAIL, called *Markovian*-GRAIL (M-GRAIL), provides the same input to the GS as in C-GRAIL, but treats goal selection as an MDP and solves it by modeling the temporal interdependency between goals as the temporal dependency between consecutive states in an MDP; it then uses Q-learning to assign a value to each goal, where values represent the long-term benefits of practicing that goal considering the intrinsic rewards that goals that depend on it may provide in the future.

4 Results

In our first experiment we compare GRAIL and C-GRAIL in a setup where the value of a contextual feature is used as precondition to determine whether the agent can activate certain spheres. In particular, spheres a, c and e can only be activated when the contextual feature (cf) is set to 1.0, while spheres b, d and f can only be activated when cs is set to 0.0. At the beginning of each trial, cf is set to 1.0 with 50% probability. While GRAIL selects tasks without considering the environmental condition, C-GRAIL receives cs as input and performs task selection as in a contextual bandit. Fig. 2 shows the performances of GRAIL and C-GRAIL on the 6 tasks during an experiment that lasts for 4000 trials. At the end of each trial, the environment is reset (all spheres are set to "off"). C-GRAIL is able to properly learn all tasks in approximately 2000 trials, while GRAIL (at the end of the simulation) has achieved high competence in only two of the tasks. This is because GRAIL performs selection (and value assignment) without taking the status of the cs into account, which is by construction important to determine whether spheres can be activated. While C-GRAIL can properly generate IMs for the different tasks only in those conditions where they can be in fact be achieved, GRAIL "wastes time" in selecting tasks even when they cannot be trained, thus impairing the learning process.

In our second experiment we introduce interdependencies between goals. In particular, sphere f can be activated only if sphere c is already active, while sphere a can be activated only when f is on. This implies that to light up sphere a, the robot has to first turn on spheres c and f. Other spheres have no dependencies. We ran simulations for 1500 epochs, each one lasting 4 trials, for a total of 6000 trials. At the end of each epoch we reset the environment; during each epoch, spheres retain their current status as determined by the actions of the robot.

Based on the first experiment we can observe that GRAIL is not capable of properly performing autonomous learning when tasks are dependent on preconditions; we thus evaluated C-GRAIL and M-GRAIL to study whether they help tackle the interdependent-task setting. Both algorithms provide as input to the goal selector, at each step, the status of the six spheres ("on" or "off"); however, they assign values to each candidate goal in different ways, as briefly discussed in Section 3. By comparing the performances of C-GRAIL and M-GRAIL (Fig. 3) we observe that while M-GRAIL reaches a perfect overall performance after about 975 epochs, C-GRAIL achieves a high performance on all goals only at the very end of the experiment.

Although both systems are capable of assigning positive values to goals only in states where their preconditions are satisfied, C-GRAIL is negatively affected whenever achieving a task such as a requires that the agent first satisfy a number of previous preconditions. Intuitively, a is "distant" from the initial condition of the system, where all spheres are off. Furthermore, whenever a task is completely learned (the robot has an optimal policy for performing a goal), the

intrinsic motivation for selecting it gradually disappears. This may lead to a situation where the robot starts selecting tasks almost at random due to the absence of intrinsic rewards, thus wasting trials in selecting goals that cannot be achieved at that moment. As a result, even though the robot may have an intrinsic motivation reward for practicing a goal (e.g. activating sphere a), it does not have intrinsic motivation for first practicing the goals that are preconditions to a; it is not, thus, capable of systematically putting the environment in the proper conditions to train a. M-GRAIL, by contrast, can rapidly learn all tasks: even when (similarly to C-GRAIL) it is no longer intrinsically motivated in achieving "simple" goals *per se* (i.e., goals with few preconditions), it ensures that the robot continues to select those goals thanks to the Q-learning algorithm, which propagates the intrinsic motivations for solving task a back to the tasks that are a's preconditions.

References

[1] B. Bakker and J. Schmidhuber. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In *Proc. of the 8-th Conf. on Intelligent Autonomous Systems*, pages 438–445, 2004.

[2] G. Baldassarre and M. Mirolli. *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer Science & Business Media, 2013.

[3] A. Baranes and P.-Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

[4] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[5] N. Duminy, S. M. Nguyen, and D. Duhaut. Learning a set of interrelated tasks by using a succession of motor policies for a socially guided intrinsically motivated learner. *Frontiers in neurorobotics*, 12, 2018.

[6] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1514–1523, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[7] S. Forestier, Y. Mollard, and P.-Y. Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[8] D. H. Grollman and O. C. Jenkins. Incremental learning of subtasks from unsegmented demonstration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 261–266. IEEE, 2010.

[9] M. B. Hafez, C. Weber, and S. Wermter. Curiosity-driven exploration enhances motor skills of continuous actor-critic learner. In *Proceedings of the 7th Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2017.

[10] M. Lopes and P.-Y. Oudeyer. The strategic student approach for life-long exploration and learning. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–8. IEEE, 2012.

[11] K. E. Merrick. Intrinsic motivation and introspection in reinforcement learning. *IEEE Transactions on Autonomous Mental Development*, 4(4):315–329, 2012.

[12] A. Mohseni-Kabir, C. Li, V. Wu, D. Miller, B. Hylak, S. Chernova, D. Berenson, C. Sidner, and C. Rich. Simultaneous learning of hierarchy and primitives for complex robot tasks. *Autonomous Robots*, pages 1–16, 2018.

[13] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[14] R. Niel and M. A. Wiering. Hierarchical reinforcement learning for playing a dynamic dungeon crawler game. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1159–1166. IEEE, 2018.

[15] P.-Y. Oudeyer, F. Kaplan, and V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(6), 2007.

[16] V. G. Santucci, G. Baldassarre, and M. Mirolli. Intrinsic motivation signals for driving the acquisition of multiple tasks: a simulated robotic study. In *Proceedings of the 12th International Conference on Cognitive Modelling (ICCM)*, 2013.

[17] V. G. Santucci, G. Baldassarre, and M. Mirolli. Which is the best intrinsic motivation signal for learning multiple skills? *Frontiers in neurorobotics*, 7:22, 2013.

[18] V. G. Santucci, G. Baldassarre, and M. Mirolli. Cumulative learning through intrinsic reinforcements. In *Evolution, Complexity and Artificial Life*, pages 107–122. Springer, 2014.

[19] V. G. Santucci, G. Baldassarre, and M. Mirolli. Grail: A goal-discovering robotic architecture for intrinsically-motivated learning. *IEEE Transactions on Cognitive and Developmental Systems*, 8(3):214–231, 2016.

[20] C. M. Vigorito and A. G. Barto. Intrinsically motivated hierarchical skill learning in structured environments. *IEEE Transactions on Autonomous Mental Development*, 2(2):132–143, 2010.

Active Domain Randomization

Bhairav Mehta
Mila, Université de Montréal
mehtabha@mila.quebec

Manfred Diaz
Mila, Université de Montréal
diazcabm@iro.umontreal.ca

Florian Golemo
Mila, Université de Montréal
golemofl@mila.quebec

Christopher Pal
Mila, Polytechnique Montréal, Element AI
christopher.pal@polymtl.ca

Liam Paull
Mila, Université de Montréal
paulll@iro.umontreal.ca

Abstract

Domain randomization is a popular technique for zero-shot domain transfer, often used in reinforcement learning when the target domain is unknown or cannot easily be used for training. In this work, we empirically examine the effects of domain randomization on agent generalization and sample complexity. Our experiments show that domain randomization may lead to suboptimal policies even in simple simulated tasks, which we attribute to the uniform sampling of environment parameters. We propose Active Domain Randomization, a novel algorithm that learns a sampling strategy of randomization parameters. Our method looks for the most informative environment variations within the given randomization ranges by leveraging the differences of policy rollouts in randomized and reference environment instances. We find that training more frequently on these proposed instances leads to faster and better agent generalization. In addition, when domain randomization and policy transfer fail, Active Domain Randomization offers more insight into the deficiencies of both the chosen parameter ranges and the learned policy, allowing for more focused debugging. Our experiments across various physics-based simulated tasks show that this enhancement leads to more robust policies, all while improving sample efficiency over previous methods.

Keywords: reinforcement learning, domain randomization, continuous control

Acknowledgements

We are grateful for the Canadian Institute For Advanced Research (CIFAR) funding that made this work possible, for the Google Cloud Platform credits that were sponsored by the AI Grant (aigrant.org).

1 Introduction

Recent trends in Deep Reinforcement Learning (DRL) exhibit a growing interest for zero-shot domain transfer, i.e. when a policy is learned in a source domain and is then tested *without finetuning* in an unseen target domain. Zero-shot transfer is particularly useful when the task in the target domain is inaccessible, complex, or expensive, such as gathering rollouts from a real-world robot. An ideal agent would learn to *generalize* across domains; it would accomplish the task without exploiting irrelevant features or deficiencies in the source domain (i.e., approximate physics in simulators), which may vary dramatically after transfer. Any agent that fails this transfer task falls prey to the *domain adaptation* problem.

One promising route to zero-shot transfer has been *domain randomization* (Tobin et al., 2017). The approach is simple: when episodically training a policy in a simulator, uniformly randomize every environment parameter of the simulation (e.g. friction, motor torque) across predefined ranges. By randomizing everything that might vary in the target environment, the hope is that eventually, the target domain will be just another variation. Yet, domain randomization is not without its flaws. Recent works suggest that the sample complexity grows exponentially in terms of the number of randomization parameters, even when dealing only with transfer between simulations (e.g., in Andrychowicz et al., 2018 Figure 8). In addition, when using domain randomization *unsuccessfully*, policy transfer fails as a black box. After a failed transfer, randomization ranges are tweaked heuristically via trial-and-error. Repeating this process iteratively, researchers are often left with arbitrary ranges that do (or do not) lead to policy convergence without any insight into how those settings may be beneficial or detrimental to the learned behavior.

In this work we investigate the impact of parameter sampling for domain randomization. We show that, in a reinforcement learning setting, uniform sampling of environment parameters is suboptimal, and moreover, that the generalization performance of the learned policy is much more sensitive to some parameters than others. This motivates the development of our algorithm, Active Domain Randomization (ADR), which has the following benefits (which we claim as our contributions):

1. ADR learns the most informative variations in the randomization space. Briefly, ADR searches for randomization settings where the agent policy deviates most from its behavior in a reference environment. We find that such environment instances correspond to harder versions of the problem and that prioritizing these samples in training leads to faster and better generalization.
2. The learned sampling strategy of ADR is reusable and can be extracted to bootstrap new agents even more efficiently while still maintaining the benefits of generalization.
3. ADR can provide insight into which dimensions and parameter ranges are most influential *before transfer*, which can alert researchers of overfitting or simulation flaws before expensive experiments are undertaken.

We illustrate how ADR is applicable to a broad spectrum of domain adaption problems, by showcasing its benefits on a variety of simulated, continuous control benchmarks.

2 Domain Randomization

Domain Randomization (DR) is a technique introduced to overcome the domain adaptation issue, especially when training policies completely in simulation and transferring them in a zero-shot manner to the real world.

DR requires a prescribed set of N_{rand} simulation parameters to randomize, as well as corresponding ranges to sample them from. This induces the notion of a *randomization space* $\Xi \subset \mathbb{R}^{N_{rand}}$, where each randomization parameter ξ_i is bounded on a closed interval $\{[\xi_{i,low}, \xi_{i,high}]\}_{i=1}^{N_{rand}}$. When a configuration $\xi \in \Xi$ is passed to a non-differentiable simulator S, it generates an environment E, which the agent policy π_θ sees and uses to train.

Generally, at the start of each episode, the parameters are uniformly sampled from the ranges, and the environment generated from those values is passed to the agent policy π_θ for training.

DR changes any to all parts of the task T's underlying Markov Decision Process (MDP)[1], with the exception of keeping R and γ constant. DR therefore generates a multitude of MDPs that are superficially similar, but can vary greatly in difficulty depending

Algorithm 1 Uniform Sampling Domain Randomization

Input: Randomization Space Ξ, Simulator S
Initialize agent policy π_θ
for each episode **do**
  Initialize trajectory buffer $\mathcal{T}_{rand}$
  // Uniformly sample parameters
  for $i = 1$ **to** N_{rand} **do**
    $\xi_i \sim U[\xi_{i,low}, \xi_{i,high}]$
    // Generate, rollout in randomized env.
    $E_i \leftarrow S(\xi_i)$
    rollout $\tau_i \sim \pi_\theta(\cdot; E_i)$
    $\mathcal{T}_{rand} \leftarrow \mathcal{T}_{rand} \cup \tau_i$
  end for
  for each gradient step **do**
    // Agent policy update
    with $\mathcal{T}_{rand}$ **update**:
      $\theta \leftarrow \theta + \nu \nabla_\theta J(\pi_\theta)$
  end for
end for

on the character of the randomization. Upon transfer to the target domain, the expectation is that policy has learned to generalize across MDPs, and sees the final domain as just another variation.

The most common instantiation of DR, Uniform-Sampling Domain Randomization (USDR) is summarized in Algorithm 1. USDR generates randomized environment instances E_i by uniformly sampling the randomization space. The agent policy π_θ is then trained on rollouts τ_i produced in those randomized environments.

3 Active Domain Randomization

3.1 Motivating Experiment

To motivate the need for a better sampling strategy, we begin by investigating the validity of the following claim: *uniformly sampling of environment parameters does not generate equally useful MDPs*. We do so by performing an experiment on a toy environment, `LunarLander-v2`, where the agent's task is to ground a lander in a designated zone, and is rewarded based on the quality of landing (fuel used, impact velocity, etc). Parameterized by an 8D state vector and actuated by a 2D continuous action space, `LunarLander-v2` has one main axis of randomization that we vary: the main engine strength (MES).

Targeting the uniform sampling strategy in DR, we aim to determine if certain environment instances (different values of the MES) are more *informative* - more efficient than others in terms of aiding learning speed and generalization. We set the total range of variation for the MES to be $[8, 20]$ (the default is 13, and lower than 7.5 makes the environment practically unsolvable when all other physics parameters are held constant) and find through simple tests that lower engine strengths generate harder MDPs to solve. Under this assumption, we show the effects of *focused domain randomization* by editing the ranges that the main engine strength is uniformly sampled from.

We train multiple agents, with the only difference between them being the randomization ranges for MES during training. The randomization ranges define what types of environments the agent sees during training.

Figure 1 shows the final generalization performance of each agent by sweeping across the entire randomization range of [8, 20] and rolling out the policy in the generated environments. We see that focusing on harder MDPs improves generalization over traditional DR, even when the evaluation environment is outside of the training distribution.

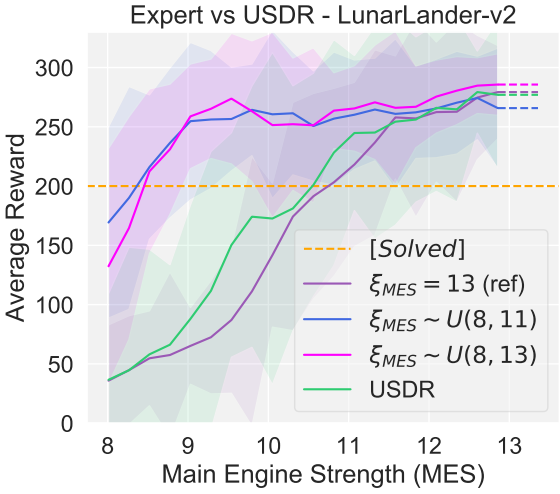


Figure 1: Generalization for various agents who saw different main engine strength ranges.

3.2 Active Domain Randomization

While interesting, the experiment in the previous section is problematic for two reasons:

1. It is rare that such intuitively *hard* MDP instances or parameter ranges are known beforehand.
2. DR is used mostly when the space of randomized parameters is high dimensional (noninterpretable).

To address these two issues, we extend our initial intuitions: not only are *all parts of a randomization range* not equally informative, but also individual *randomization dimensions* are not equally useful to focus on. In `LunarLander-v2`, the main engine strength is more crucial to task performance than the side engine strength, which only marginally affects generalization performance when varied. An ideal randomization scheme would learn this inherent difference between the dimensions' values and focus on the most difficult environment instances rather than sampling uniformly across the entire space.

To this end we propose ADR, summarized in Algorithm 2. ADR tries to find these informative MDPs within the randomization space by formulating the search as an Reinforcement Learning (RL) problem. ADR attempts to learn a policy μ_ϕ where the states are proposed randomization configurations $\xi \in \Xi$ and actions are changes to those parameters.

We learn a discriminator-based reward for μ_ϕ, similar to the one originally proposed in Eysenbach et al. (2018):

$$r_D = \log D_\psi(y|\tau_i \sim \pi_\theta(\cdot; E_i)) \tag{1}$$

[1]The effects of DR on action space A are usually implicit or are carried out on the simulation side.

where y is an boolean variable denoting the discriminator's prediction of which type of environment (a randomized environment E_i or reference environment E_{ref}) the trajectory τ_i was generated from. In our work, we define the E_{ref} to be the environment defined by the *default* parameter configuration ξ_{ref}, which comes along with the original task definition. Intuitively, we reward the policy μ_ϕ for finding regions of the randomization space that produce environment instances where the *same* agent policy π_θ acts differently than in the reference environment. The agent policy π_θ sees and trains *only* on the randomized environments (as it would in traditional DR), using the environment's task-specific reward for updates. As the agent improves on the proposed, problematic environments, it becomes more difficult to differentiate whether any given state transition was generated from the reference or randomized environment. Thus, ADR can find what parts of the randomization space the agent is currently performing poorly on, and can *actively* update its sampling strategy throughout the training process.

To encourage sufficient exploration in high-dimensional randomization spaces, we parameterize μ_ϕ with Stein Variational Policy Gradient (SVPG). SVPG benefits from multiple, diverse policies and gives the agent policy π_θ the same type of environment variety seen in DR while still using the learned reward to hone in on problematic MDP instances. In addition, SVPG is highly parallelizable, which allows us to gather agent rollouts across various randomized environments simultaneously (Lines 7 to 18 of Algorithm 2).

Algorithm 2 Active Domain Randomization

Input: Ξ: randomization space, S: simulator, ξ_{ref}: reference parameters
Initialize π_θ: agent policy, μ_ϕ: SVPG particles, D_ψ: discriminator, $E_{ref} \leftarrow S(\xi_{ref})$: reference environment
while not $max\_timesteps$ **do**
  for each sampling step **do**
    rollout $\xi_i \sim \mu_\phi(\cdot)$
  end for
  for each ξ_i **do**
    // Generate, rollout in randomized env.
    $E_i \leftarrow S(\xi_i)$
    rollout $\tau_i \sim \pi_\theta(\cdot; E_i)$, $\tau_{ref} \sim \pi_\theta(\cdot; E_{ref})$
    $\mathcal{T}_{rand} \leftarrow \mathcal{T}_{rand} \cup \tau_i$
    $\mathcal{T}_{ref} \leftarrow \mathcal{T}_{ref} \cup \tau_{ref}$
    for each gradient step **do**
      // Agent policy update
      with $\mathcal{T}_{rand}$ **update**:
        $\theta \leftarrow \theta + \nu \nabla_\theta J(\pi_\theta)$
    end for
  end for
  // Calculate reward for each proposed environment
  for each $\tau_i \in \mathcal{T}_{rand}$ **do**
    Calculate reward with associated ξ_i and E_i using Eq. (1)
  end for
  // Update randomization sampling strategy
  for each particle μ_{ϕ_i} **do**
    Update particles using SVPG
  end for
  // Update discriminator
  for each gradient step **do**
    Update D_ψ with τ_i and τ_{ref} using SGD.
  end for
end while

4 Results

4.1 Implementation Details

To test ADR, we experiment on `LunarLander-v2`[2], a 2 degrees of freedom (DoF) environment in which the agent has to softly land a spacecraft, implemented in Box2D (detailed in Section 3.2). Across all experiments, our agent policy is trained with Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2015), although the agent policy can be substituted with any other on or off-policy algorithm with only minor changes to Algorithm 2. Our discriminator-based reward generator is a three layers, 128-neurons, Fully Connected Neural Network (FCN).

The simulator parameter sampling strategy is parameterized by SVPG with RBF Kernel and temperature $\alpha = 10$, and we use Advantage Actor-Critic (A2C) to calculate unbiased and low variance gradient estimates. Each of the SVPG particles is also an FCN of two layers of 100 neurons each. All experiments results are plotted (mean) averaged across five seeds, five trials per evaluation point, with one standard deviation shown.

4.2 Toy Experiments

To investigate whether ADR's learned sampling strategy provides a tangible benefit in both agent generalization and learning speed, we start by comparing it against traditional DR (labeled as USDR) on `LunarLander-v2` and vary only the main engine strength (MES). In Figure 3, we see that ADR approaches expert-levels of generalization whereas traditional DR fails to generalize on lower MES ranges.

Figure 2 explains the flexibility of ADR by showing generalization and sampling distribution at various stages of training. ADR starts by sampling approximately uniform for the first 650K steps, when it is apparent that it finds a deficiency in the policy on higher ranges of the MES. As those areas become more frequently sample between steps 650K-800K steps, the agent learns to solve all of the higher-MES environments, as shown by the generalization curve for 800K steps. As a result, the discriminator is no longer able to differentiate reference and randomized trajectories from the higher MES regions, and starts to reward environment instances generated in the lower end of the MES range, which improves generalization by the completion of training.

[2] https://gym.openai.com/envs/LunarLander-v2/

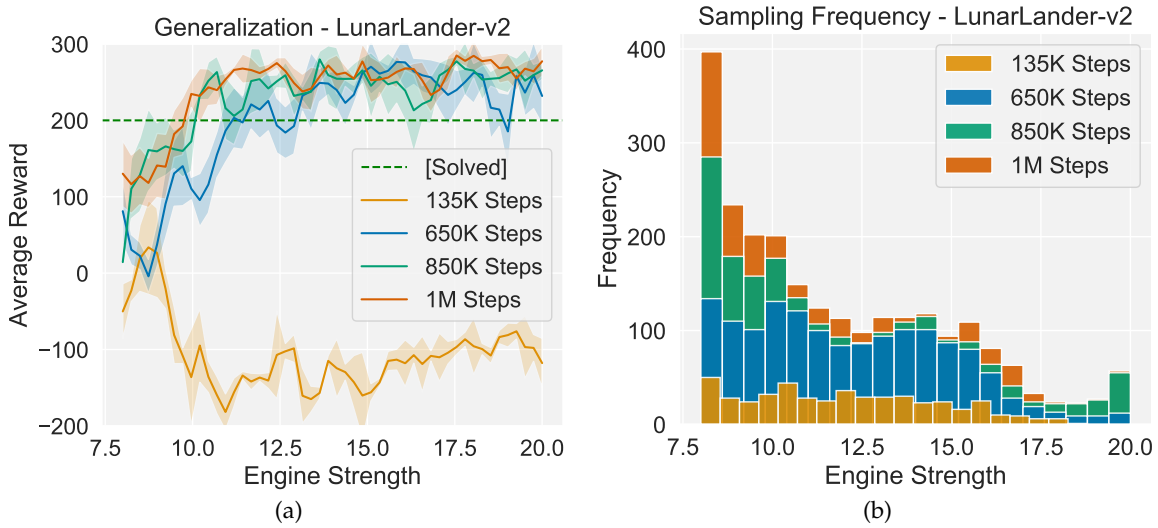


Figure 2: Agent generalization (a) and environment sampling frequency (b) throughout training on `LunarLander-v2`.

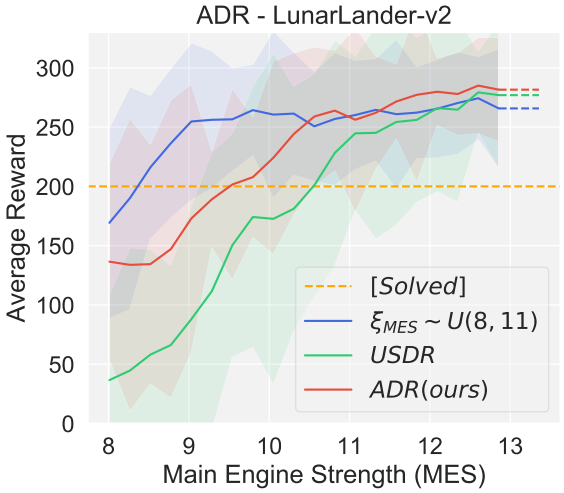


Figure 3: Generalization on `LunarLander-v2` for an expert interval selection, ADR, and USDR. Higher is better.

References

Andrychowicz, Marcin et al. (2018). "Learning dexterous in-hand manipulation." In: *arXiv preprint arXiv:1808.00177*.

Eysenbach, Benjamin et al. (2018). "Diversity is All You Need: Learning Skills without a Reward Function." In: *arXiv preprint arXiv:1802.06070*.

Lillicrap, Timothy P et al. (2015). "Continuous control with deep reinforcement learning." In: *arXiv preprint arXiv:1509.02971*.

Tobin, Josh et al. (2017). "Domain randomization for transferring deep neural networks from simulation to the real world." In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, pp. 23–30.

Perception as prediction using general value functions in autonomous driving applications

Daniel Graves
Huawei Noah's Ark Lab
Huawei Technologies Canada, Ltd.
Edmonton, AB, Canada
daniel.graves@huawei.com

Kasra Rezaee
Huawei Noah's Ark Lab
Huawei Technologies Canada, Ltd.
Markham, ON, Canada
kasra.rezaee@huawei.com

Sean Scheideman
Department of Computer Science
University of Alberta
Edmonton, AB, Canada
searn@ualberta.ca

Abstract

Autonomous driving is a challenging domain for control due to the vast scenario and environment complexities that an agent will face. Designing a reward signal to learn end-to-end is not an easy task. In this work, we propose and investigate a perception as prediction framework as an alternative perspective in autonomous vehicle following the Horde framework. We investigate how to learn and use policy-based predictions of safety and speed in the problem of adaptive cruise control (ACC) while also learning to predict safety from being rear-ended by other agents. We train the predictors in three different simulation environments and design a few hand-crafted controllers to compare with an LQR-based baseline in challenging ACC scenarios. We then do sim2real transfer of the predictions to the a Clearpath robot and a Lincoln MKZ vehicle and demonstrate that the predictions can be used in the real-world.

Keywords: general value functions, prediction learning, autonomous driving

1 Introduction

Understanding the world by learning predictions and using those predictions to act intelligently in the world is becoming an important topic of research, cf [1][2][3][4]. Modern theory of the brain shows that we are predictive machines that constantly try to match incoming sensory inputs with predictions [5]. The Horde framework embraces this idea of predicting sensorimotor signals [1] using general value functions to learn and make action and policy dependent predictions. There have been a number of successful applications of GVFs including controlling a myoelectric robotic arm prosthesis [2] and controlling a laser welding robot from a camera [3].

In this work, we are interested in exploring a perception as prediction architecture in autonomous driving. A common approach to designing an autonomous vehicle is to build layers to abstract the decision making that happens in planning and controlling a vehicle [6]. The planning layers use the world model generated by perception tasks to decide on a route, choose behaviors such as when to change lanes and then plan a path in the world. This is then passed to a control layer that executes the plan produced by the planning layers including lateral (steering) and longitudinal control (throttle and brake). Motivated by [5], we propose augmenting this architecture with action-oriented predictions where the key difference is that the predictions provide a link between the agent's actions and the sensor data. This is an important distinction because it enables the autonomous agent to understand how the actions taken affect the sensor readings as well as the objects detected by a traditional perception pipeline. This information is very helpful in allowing an agent to understand what actions to take in order to achieve a variety of goals in the environment.

We demonstrate in our experiments that there is one target policy that is very useful in autonomous driving, and specifically adaptive cruise control, and that is "what will happen if I keep doing what I'm doing?" We believe this policy is especially interesting in problems like autonomous driving where actions tend to change slowly over time and it provides a signal that a controller can use to adjust its action.

2 Predicting Safety with GVFs

We borrow from a classical definition of safety in autonomous driving and adaptive cruise control using the inter-vehicle distance model also called headway [7]. However, we extend this model to three dimensions to allow for use with high dimensional LIDAR sensors as well as low dimensional radar sensors. The safety function is the pseudo-reward signal

(also called the cumulant in general value function literature) which the predictor must learn to predict. The safety cumulant function c_f maps the current state s_t at time t to a ego safety score on the interval $[0, 1]$. There are two safety zones in our implementation: ego safety (or front safety) and rear safety. The definition of front safety used is

$$c_f(s_t) = \begin{cases} 0, & \text{if } n_f > \beta_f. \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

where n_f is the number of points returned by either LIDAR or radar sensors that are inside the front safety zone (or box) and $\beta_f \geq 0$ is a minimum threshold. The width of the front safety zone is the width of the vehicle and the height is the height of the vehicle. The length of the front safety zone is the headway to the vehicle in front and is proportional to the vehicle's speed following the inter-vehicle distance model $h_f = d_{\text{min}} + v\tau$ where τ is the desired headway in seconds, d_{min} is the stand-still distance, and v is the current speed of the vehicle. For rear safety, we build a three dimensional safety zone for the rear vehicle and calculate $h_r = d_{\text{min}} + v_r\tau$ where v_r is the speed of the rear vehicle.

The safety cumulants are the signals that are predicted. We use a similar approach as [1] only instead of using the GQ(λ) algorithm to learn a general value function, we use TD($\lambda = 0$) to learn a general value function. In addition, the focus in this work will be on learning general action value functions which may be more useful for control, where the prediction is a function of state and action; thus, the predictive question is "will I be safe if I take action a and take similar actions thereafter?" The reason is that the predictive question can be viewed as a kind of predictive model of the world that permits queries over a set of possible next actions.

The goal is to learn an estimator that predicts the return of the cumulant G_t defined by

$$G_t \equiv \sum_{k=0}^{\infty} (\prod_{j=0}^{k} \gamma_{t+j+1}) c_{t+k+1} \tag{2}$$

where $0 \leq \gamma_t < 1$ is the continuation function and c_t is the cumulant (pseudo-reward) sampled at time t. The general value function is defined as $Q^\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a, a_{t+1:T-1} \sim \pi, T \sim \gamma]$ where π, γ, and c make up the predictive question [1]. Each GVF Q^π is approximated with an artificial neural network parameterized by θ denoted as $\hat{q}^\pi(s, a, \theta)$.

Using non-linear function approximation introduces potential challenges in learning because there is no proof of convergence. In addition, off-policy learning where the target policy π may be different from the behavior policy μ could be problematic with deep neural networks if importance sampling ratios are required.

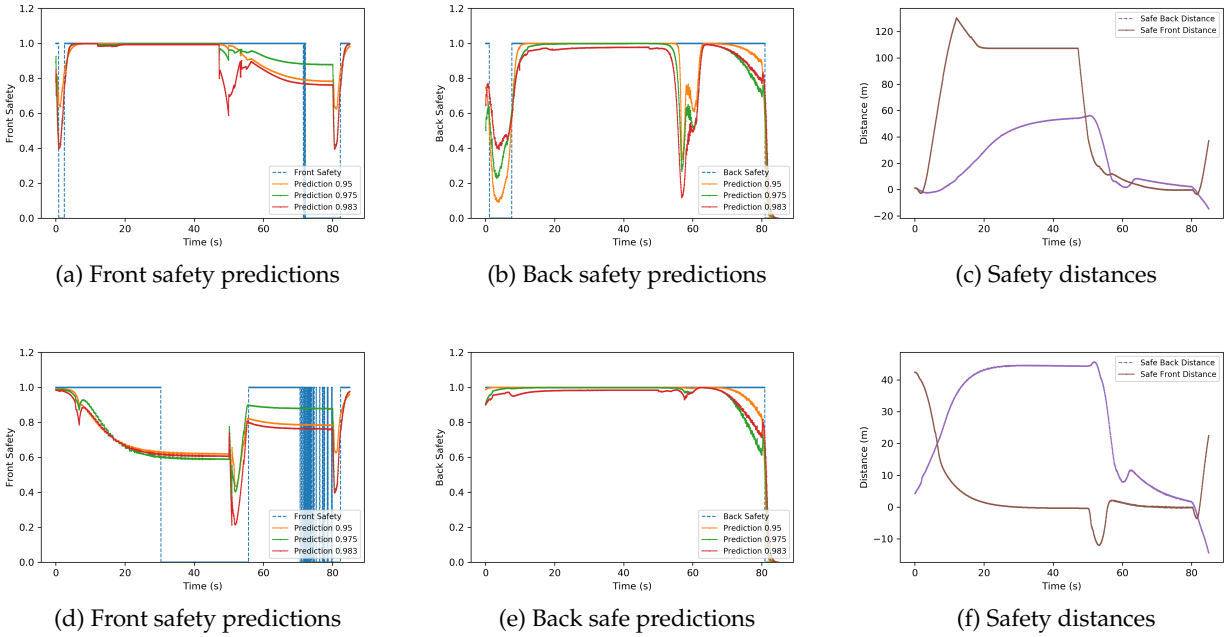
The approach adopted here uses TD($\lambda = 0$) [8] to learn the predictions using non-linear function approximation with an experience replay buffer. The loss for the general value function $\hat{q}^\pi(s, a, \theta)$ is the squared TD error: $L(\theta) = \mathbb{E}_{s\sim d_\mu, a\sim\mu}[(y - \hat{q}^\pi(s, a, \theta))^2]$ where the target y is produced by bootstrapping a prediction of the value of the next state and action taken from the target policy π given by $y = \mathbb{E}_{s'\sim P, a'\sim\pi}[c + \gamma\hat{q}^\pi(s', a', \theta)]$ where y is a bootstrap prediction using the most recent parameters θ but is ignored in the gradient descent. d_μ is the state distribution of the behavior policy μ and P is the Markovian transition distribution over next state s' given state s and action a. The time subscript on c and γ has been dropped to simplify notation. If the agent behaves with the same policy as the target policy π such that $\mu(a|s) = \pi(a|s)$ then the approach is on-policy learning otherwise it is off-policy. Note that this approach doesn't correct for the state distribution d_μ because it still depends on the behavior policy μ. Both on-policy and off-policy approaches were tried but since the behavior policy constructed to achieve suitable exploration was very similar to the target policy, no appreciable difference was noticed.

When the agent observes a state s and chooses an action a according to its behavior policy, it receives cumulant c, continuation γ and observes next state s' and store this tuple in the replay buffer. We also generate an action $a' \sim \pi$ and store it in the replay buffer whether that is the next action taken (on-policy) or not (off-policy). When training the GVF, a mini-batch of $m < n$ samples, where n is the size of the replay buffer, is sampled randomly to update the parameters θ. The updates can be either accomplished on-line, while collecting and storing experience in the replay buffer, or off-line after the data has been collected since the target policy doesn't change.

In this work, we learn separate estimators: one for each cumulant of future front safety, rear safety, and speed. The cumulants are scaled by a factor of $1 - \gamma_{t+1}$ to normalize them; this ensures the predicted safety is on the interval $[0, 1]$ since the sum of an infinite geometric series of $0 \leq \gamma < 1$ is $1/(1 - \gamma)$ if γ is constant.

3 Experiments

An analysis of the predictions and the behavior of the controllers that use these predictions are provided in the TORCS environment under a number of different scenarios. After evaluation, the rule-based controller was selected for imple-



(a) Front safety predictions (b) Back safety predictions (c) Safety distances

(d) Front safety predictions (e) Back safe predictions (f) Safety distances

Figure 1: Predicting front and rear safety with different values of γ without using the predictions for control. Top row (a)-(c) is from the emergency stop scenario and the bottom row (d)-(f) is from the follow-and-stop scenario.

mentation on the Jackal robot and the autonomous vehicle platform since it performed similarly with the fuzzy controller and was simpler to tune. A two stage approach is used: (1) learn the predictors by following the predictor's target policy and (2) use the predictors in autonomous driving applications (such as a warning system or adaptive cruise control).

The target policy chosen was the normal distribution centered on the last action taken, i.e. $\pi(a_t|s_t, a_{t-1}) = \pi(a_t|a_{t_1}) = \mathcal{N}(a_{t-1}, \sigma^2)$ where σ is a tunable parameter. This target policy represents the question "what if I keep doing what I'm doing?" The behavior policy chosen was a Wiener process where the next action is the last action plus noise generated by a normal distribution centered on the last action taken and with standard deviation equal to σ for our target policy. However, in order to facilitate exploration of the state and action space, the agent occasionally interrupts the Wiener process and chooses a random action according to uniform probability and then continues with the random walk.

We trained predictions for γ values of 0.95, 0.975 and 0.983. When training the safety predictors, the other vehicles are controlled by a similar random walk process. Therefore, the training terminates with $\gamma = 0$ upon collision with another vehicle.

3.1 TORCS Experiments

Here we highlight the two challenging high-speed scenarios: (a) emergency stop and (b) follow-and-stop. The target speed in both of these experiments was 100 km/h. In the first scenario the vehicle approaches a stopped vehicle and must stop quickly to avoid collision. In the second scenario the vehicle follows a slower vehicle going 80 km/h which then abruptly stops requiring the vehicle to react and slow down without a collision. We compared against an LQR-based controller [7] called ACC/CA as a baseline. The target safety parameters for all controllers were defined by a desired spacing of $\tau = 3$ seconds and minimum stopping distance of $d_{\text{min}} = 4$ meters. The objective was not to beat the baseline but to match its performance and demonstrate that perception as prediction is a viable approach in autonomous driving that deserves more attention and further research.

In the first set of experiments, the performance of the predictions is analyzed by comparing them with the safety signals being predicted. The ACC/CA baseline controller is used to drive the vehicle.

The safe distances for front and rear are shown in figure 1(c) and (f) for reference. The safety GVFs predict both front and rear safety effectively and anticipate when the safety could change based on observations of the other agents. For example, in figure 1(b) where the back safety dips at around $t = 58$ s, the vehicle is slowing down very quickly while the vehicle behind is not reacting fast enough. Once the vehicle behind starts to decelerate sufficiently, the predictions jump back up again predicting that the vehicle is safe from a rear-end collision.

When the predictions are applied to the adaptive cruise control problem, we implemented rule-based and fuzzy-based controllers and they perform similarly which suggests the predictions learned are potentially useful for a variety of

controllers. The GVF-based controllers appeared to optimize the safety rather well for all values of gamma with $\gamma = 0.983$ particularly tending to keep the vehicle safer than ACC/CA during deceleration possibly because of being longer-term predictions. In fact, in most of our experiments, the GVF-based controllers did not cross the safety distance threshold.

In terms of computational complexity, the GVF-based approach with the fuzzy controller required only needed to search across 21 possible next actions to determine a suitable next action. The reason is because the predictions are policy-based rather than action trajectory-based which otherwise would have required a significantly larger search space over all possible action sequences. We therefore argue that policy-based predictions are a viable way achieve predictive control with appropriately selected target policies while keeping computational requirements low.

3.2 Demonstrating in the Real-World

We then trained front safety predictions using a deep convolutional neural network in Gazebo to test on a real-world Clearpath Jackal robot. In the training environment, randomly shaped geometric objects or were generated in the scene to help with sim-to-real transfer to the real-world. We then tested the predictors on the Clearpath Jackal robot. The robot has a Hokuyo UTM 30LX laser range finder which produces 1040 distance measurements in a 260° arc in front of the robot. A 5MP front facing color camera was used to follow blue tape for lateral steering control using a rule-based controller for longitudinal control. We used a deep neural network with 6 convolutional layers and 2 fully connected layers to predict safety from a history of 3 LIDAR measurements and the vehicle's current speed v_t. The safety parameters $\tau = 1.5$ seconds and $d_{\text{min}} = 0.4$ meters were used since the robot can stop fairly quickly.

The safety predictors were tested on a real robot where we tried several different situations: (a) following a human with varying walking speeds along a pre-defined path, (b) approaching a stationary obstacle, and (c) reacting to a person walking in front of the robot suddenly. In all cases, the robot was able to stop without collision.

We also trained front safety predictors in the Webots simulator environment for deploying on an autonomous vehicle in a controlled test environment. The objects detected in the scene were supplied as input to the neural network which included the distance and speed of the vehicle in front and in the same lane. We tested a similar rule-based controller as used on the Jackal robot and was able to achieve the desired following behavior. An emergency brake test was performed where both virtual and real objects were placed in the scene of the vehicle requiring it to stop immediately. Finally, we proceeded to test the vehicle in a large circular road where the vehicle had to stop for pedestrians and other vehicles. The performance was often comfortable and the speed control usually felt human-like. The tests showed that the vehicle was still able to respond quickly and safely in situations that required emergency braking as well as follow another vehicle smoothly.

References

[1] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *10th Int. Conf. on Autonomous Agents and Multiagent Systems - Vol. 2*, ser. AAMAS '11, 2011, pp. 761–768.

[2] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton, "Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning," in *2011 IEEE Int. Conf. on Rehabilitation Robotics*, June 2011, pp. 1–7.

[3] J. Gunther, P. M. Pilarski, G. Helfrich, H. Shen, and K. Diepold, "Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning," *Mechatronics*, vol. 34, pp. 1 – 11, 2016.

[4] G. Kahn, A. Villaflor, P. Abbeel, and S. Levine, "Composable action-conditioned predictors: Flexible off-policy learning for robot navigation," *CoRR*, vol. abs/1810.07167, 2018.

[5] A. Clark, "Whatever next? predictive brains, situated agents, and the future of cognitive science," *Behavioral and Brain Science*, vol. 36, no. 3, pp. 181–204, 2013.

[6] W. Zong, C. Zhang, Z. Wang, J. Zhu, and Q. Chen, "Architecture design and implementation of an autonomous vehicle," *IEEE Access*, vol. 6, pp. 21 956–21 970, 2018.

[7] S. Moon, I. Moon, and K. Yi, "Design, tuning, and evaluation of a full-range adaptive cruise control system with collision avoidance," *Control Engineering Practice*, vol. 17, no. 4, pp. 442 – 455, 2009.

[8] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

Optimal nudging

Mathew Hardy
Department of Psychology, Princeton University
mdhardy@princeton.edu

Frederick Callaway
Department of Psychology, Princeton University
fredcallaway@princeton.edu

Thomas L. Griffiths
Departments of Psychology and Computer Science, Princeton University
tomg@princeton.edu

Abstract

People often face decisions where errors are costly but computing the optimal choice is intractable or prohibitively difficult. To address this, researchers have developed nudge theory as a way to lead people to better options without imposing restrictions on their freedom of choice. While heuristics and case-by-case evaluations are usually used to predict and explain nudges' effects on choice, another way of interpreting these effects is that nudges can change the costs of attaining certain pieces of information. These changes in costs then bias people towards or away from making particular choices. In this paper, we propose a method for predicting the effects of choice architecture on option selection by modeling deliberation as a metalevel Markov decision process and nudging as the reduction of certain computational costs. This allows us to construct *optimal nudges* by choosing cost modifications to maximize some objective function. This approach is flexible and can be adapted to arbitrary decision making problems. Furthermore, by making the objectives of nudging explicit, the approach can address ethical concerns regarding the effects of nudging and the role people should have in choosing how, when, and why they are nudged. We demonstrate the strength of this framework by applying it to the Mouselab paradigm, where deliberation costs are made explicit. We find that a version of our approach leads to significantly higher participant reward, both increasing the quality of their choices and lowering the cost of making these choices.

Keywords: decision-making; bounded rationality; resource-rationality; decision-support

Acknowledgements

This research was supported by a grant from Facebook Reality Labs

1 Introduction

Every day people encounter a seemingly neverending set of complicated decisions and difficult tradeoffs. When facing these dilemmas, people often make choices that deviate from classical notions of rationality (Kahneman, Slovic, & Tversky, 1982). These deviations, made in both small and large-scale decisions, are often costly to both individuals and society (Kahneman et al., 1982). Policy makers have traditionally sought to reduce these costs through educational programs, incentives, and mandates designed to improve people's choices (Benartzi et al., 2017; Thaler & Sunstein, 2008). However, another increasingly popular approach is to change the context and structure of the decisions people encounter, rather than trying to change behavior directly. These types of modifications, often called "nudges", change only the architecture of a decision, and include changing default options, information availability, and social norms. While often subtle, nudges have been shown to lead to increases in retirement savings, reductions in energy consumption, and increased vaccination rates, among other things (Benartzi et al., 2017). Furthermore, the cost of administrating nudges is often substantially lower than organizing effective educational campaigns or new incentive structures (Benartzi et al., 2017).

However, nudges can be controversial. There is often disagreement about what the effects of a change in choice architecture should be (Goodwin, 2012), and nudges can influence behavior in ways that people are unaware of, cannot control, or that benefit other parties at the expense of their own well-being. Furthermore, the domains in which nudging have been most successful are somewhat limited, with most successful applications the result of domestic policy or marketing.

In this paper we present a formal framework for *optimal nudging*. Using metalevel Markov decision processes, we characterize nudging as the targeted reduction of computational costs leading to a desired change in a decision maker's reasoning process and/or choices. We apply and test our model in a modified version of the Mouselab paradigm (Payne, Bettman, & Johnson, 1988), in which cognitive costs are represented as the monetary costs of uncovering information about options' possible outcomes. We conclude by discussing implications of our approach, and ways in which it can be used to address some of the ethical concerns regarding nudging.

2 Modeling optimal nudging

2.1 Metalevel Markov decision processes

We model the decision maker's deliberation process as a metalevel MDP (Hay, Russell, Tolpin, & Shimony, 2012). A metalevel MDP treats reasoning as a sequential decision problem. Formally a metalevel MDP is defined analogously to a standard undiscounted MDP, $(\mathcal{B}, \mathcal{C}, T_{\text{meta}}, r_{\text{meta}})$, where the states, $\mathcal{B}$, correspond to beliefs and the actions, $\mathcal{C}$, correspond to cognitive operations, or computations. The effect of computations on beliefs is described in the metalevel transition function T_{meta}, and the costs and benefits of computations are described in the reward function, r_{meta}. Finally, a reasoning strategy is formalized by a metalevel policy, $\pi_{\text{meta}} : \mathcal{B} \to \Delta(\mathcal{C})$, which selects computations to perform in a given belief state. Note that $\Delta(\cdot)$ denotes the set of all distributions over a set.

A belief $b \in \mathcal{B}$ is a distribution over some parameters of the environment, θ, that determine the reward, $r(a; \theta)$, of each possible physical action, $a \in \mathcal{A}$. At any moment, the metalevel policy can choose to cease deliberation by executing the termination operation, $\perp$. This passes control to an action-selecting "object-level" policy, π_{act}, that selects an action uniformly from those that are optimal in the final belief state: $\pi_{\text{act}}(a \mid b) = \text{Uniform}(a; \operatorname{argmax}_{a' \in \mathcal{A}} \mathbb{E}_{\theta \sim b}[r(a'; \theta)])$. The metalevel reward for terminating computation is the external reward, r, for the chosen action, a, under the *true* parameters, θ^*; that is, $r_{\text{meta}}(b, \perp) = r(a; \theta^*)$. In general, this value will be higher the more accurate the belief b is, rewarding belief-refining computations. However, computation comes at a cost: $r_{\text{meta}}(b, c)$ is strictly non-positive (and typically negative) for all computations besides the termination operation, $\perp$. This sets up a natural tradeoff between the benefits and costs of computation. At each time point, the metalevel policy must determine which computations (if any) will lead to sufficient improvement in expected decision quality so as to outweigh their costs.

2.2 Optimal nudging as computational cost modification

The primary contribution of this paper is a formal approach to nudging based on the idea that some nudges modify the costs of the computations available to a decision maker, thus changing the sequence of computations—and ultimately, the decision—she is likely to make. Assume that we can model some decision-making problem as a metalevel MDP $(\mathcal{B}, \mathcal{C}, T_{\text{meta}}, r_{\text{meta}})$, and let $r_{\text{meta}}(b, c) = \lambda_c$ for all $c \in \mathcal{C} \setminus \{\perp\}$, where $\boldsymbol{\lambda}$ is a vector giving the cost of all available computations (excluding the termination operation, $\perp$). We can then formalize nudging as replacing the original costs $\boldsymbol{\lambda}$ with modified costs $\tilde{\boldsymbol{\lambda}}$. *Optimal* nudging is thus the selection of $\tilde{\boldsymbol{\lambda}}$ to maximize some objective function.

The choice of a suitable objective function depends on the goals one wishes to accomplish through nudging. For example, many nudges aim to maximize the probability that people take a certain action, such as those seeking to maximize organ donation or recycling rates. This kind of goal can be formalized as maximizing the probability of the decision maker

choosing a specific action,

$$p(a \mid \tilde{\boldsymbol{\lambda}}; \theta^*, \pi_{\text{meta}}) = \mathbb{E}\big[\pi_{\text{act}}(a \mid B_\perp) \mid \tilde{\boldsymbol{\lambda}}, \theta^*, \pi_{\text{meta}}\big] = \int_{B_T \in \mathcal{B}} p(B_T \mid \tilde{\boldsymbol{\lambda}}, \theta^*, \pi_{\text{meta}})\, \pi_{\text{act}}(a \mid B_T)\, dB_T\,, \tag{1}$$

where $p(B_T \mid \tilde{\boldsymbol{\lambda}}, \theta^*, \pi_{\text{meta}})$ gives the probability that a certain belief will be the final belief if the metalevel policy π_{meta} chooses computations in a metalevel MDP with costs $\tilde{\boldsymbol{\lambda}}$ and true parameters θ^*. Note that we are implicitly conditioning on the full metalevel MDP, but we leave it out of the equations to ease notational burden.

Other nudges do not aim to make people choose a specific option, but rather to improve the overall quality of their decisions, encouraging them, for example, to make healthier eating choices or choose more diversified investment portfolios. We can model this kind of goal as maximizing the expected utility of the decision maker's choice,

$$U(\tilde{\boldsymbol{\lambda}}; \theta^*, \pi_{\text{meta}}) = \sum_{a \in \mathcal{A}} p(a \mid \tilde{\boldsymbol{\lambda}}; \theta^*, \pi_{\text{meta}})\, r(a)\,. \tag{2}$$

A potential limitation of the utility-maximizing approach is that it ignores the cost of deliberation (Griffiths, Lieder, & Goodman, 2015). In some cases, we might want to not only encourage people to make *better* decisions, but also to make it *easier* to make those decisions. We can formalize this goal as minimizing the total expected computational cost to the decision maker,

$$l(\tilde{\boldsymbol{\lambda}}; \theta^*, \pi_{\text{meta}}) = \mathbb{E}\bigg[\sum_{t}^{T-1} \tilde{\lambda}_{C_t} \Big| \tilde{\boldsymbol{\lambda}}, \theta^*, \pi_{\text{meta}}\bigg]\,. \tag{3}$$

In addition to optimizing with respect to a single one of the above equations, we might want to optimize a combination. In particular, we can combine expected decision utility (Equation 2) and expected deliberation cost (Equation 3) into a single description of how "well-off" the decision maker is,

$$g(\tilde{\boldsymbol{\lambda}}; \theta^*, \pi_{\text{meta}}) = U(\tilde{\boldsymbol{\lambda}}; \theta^*, \pi_{\text{meta}}) - l(\tilde{\boldsymbol{\lambda}}; \theta^*, \pi_{\text{meta}}) = \mathbb{E}\bigg[\sum_{t}^{T} r(B_t, C_t) \Big| \tilde{\boldsymbol{\lambda}}, \theta^*, \pi_{\text{meta}}\bigg]\,. \tag{4}$$

Combining the decision utility and deliberation cost in this way, we recover the standard objective function in a metalevel MDP—maximizing the sum of metalevel rewards, or the *metalevel return*. However, rather than optimizing this objective through our choice of a metalevel policy, we aim to optimize the metalevel return by modifying the problem itself, in particular, by reducing the costs of certain computations.

The metalevel return can trivially be optimized by setting the cost of all computations to zero. In this case, the decision maker will always take all possible computations, making the best possible decision while paying no cost. However, it is unlikely that completely eliminating deliberation costs is possible. Thus, we assume that the modifications are constrained by a budget, Z. That is, $\sum_c \lambda_c - \tilde{\lambda}_c < Z$. We impose the additional constraint that costs cannot be increased or driven below zero.

Although any combination of the objectives defined above are possible, we will focus on the metalevel return, defined in Equation 4. Critically, however, our approach can be trivially extended to optimize any combination of the functions defined above, and indeed any function of a sequence of computations followed by a choice.

To estimate any of these objectives, we must make an assumption about the decision maker's reasoning strategy, π_{meta}. One principled choice is to assume that the decision maker is metalevel optimal and maximizes the expected sum of metalevel rewards. This approach has been found to model human behavior somewhat well in simple decision making (Gul, Krueger, Callaway, Griffiths, & Lieder, 2018) and planning (Callaway et al., 2018) problems. However, in more complex domains, even approximating the optimal metalevel policy is computationally intensive. Thus, this approach may not be desirable from an implementation perspective. As an alternative, we could assume that the decision maker employs some other metalevel policy that is adaptive but not optimal. For example, the meta-greedy policy (Russell & Wefald, 1991) acts myopically, choosing each computation as if it were the last one. This policy often behaves similarly to the optimal policy, while being easy to compute. Thus, we assume that π_{meta} is the meta-greedy policy, and use it to determine the optimal cost modifications in the experiment described below.

3 Experiment: testing optimal nudging

We evaluated the proposed optimal nudging method in a modified version of the Mouselab paradigm (Payne et al., 1988). In this setup, participants have to make choices between different options with known and unknown payoff values. The paradigm externalizes computations as information-gathering operations (clicks) that reveal these values, computational cost as the monetary cost of these operations, and belief states as configurations of revealed and hidden values. By using such a paradigm, we can more easily make assumptions about the metalevel MDP underlying the participants' decisions, thus allowing us to test our cost-modification approach directly.

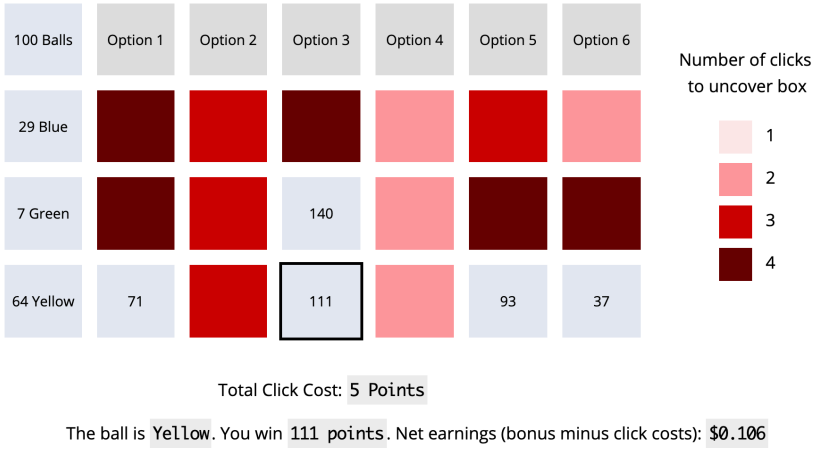


Figure 1: Experimental interface. On every trial, participants made a choice between six options. After choosing an option, a single ball color (blue, green, or yellow) was selected with percentage probability equal to the number of balls of that color. The option then paid out with the value indicated by the corresponding cell. The values in some cells were shown immediately, while others were hidden at trial onset. Participants could click on these cells to reveal the values, paying one point for each click. The number of clicks required to reveal each cell was indicated by its color.

3.1 Methods

An example of the experimental interface is given in Figure 1. Participants chose between six options (columns), each with three possible payoff values (rows). After making a choice, a ball was drawn from a simulated lottery machine with 100 balls, and the chosen option paid out depending on the color of the drawn ball. The percentage probability that a certain color ball was drawn was simply the number of balls indicated in the far left column. Different options paid different values depending on which color ball was drawn; some of these values were revealed at trial onset, while others were hidden. To reveal a hidden value, participants had to click the value they wished to reveal between one and four times (see Figure 1), paying one point for each click. These clicks correspond to computations in the metalevel MDP. The cost to reveal each cell was sampled uniformly from $\{0, 1, 2, 3, 4\}$ to mask the cost-reductions (described below). Cell values were sampled from a normal distribution with a mean of 75 points and a standard deviation 36 points (truncated at 0 points).

On each trial, the cost structure was modified according to either the proposed optimal nudging method or a random baseline. In both cases, the cost-modification budget was set to $Z = 6$ and the space of budget allocations was constrained to uniform reductions of 2 or 3 costs (each cost being reduced by 3 or 2 points respectively). Optimal costs were chosen to maximize the metalevel return of the meta-greedy policy (Equation 4). In particular, we employed a greedy search algorithm that made local adjustments to $\tilde{\boldsymbol{\lambda}}$ until no adjustment could further improve the metalevel return. The random cost modification was determined by randomly sampling three costs and reducing each by 2 points.

We recruited 150 participants from Mechanical Turk. Participants first completed a practice trial, and then 20 test trials in which 10 problems had random sales and 10 had optimal sales. Each participant completed the same set of 21 problems, but problem order and each problem's modification type varied randomly between participants.[1] At the end of the game, participants' total points were paid as a bonus with 10 points equal to 1 cent. Participants earned \$0.25 for participating in the study plus an average bonus of \$1.71.

3.2 Results

On average, participants earned 81.55 points on trials with random modifications and 89.66 points on trials with optimal modifications (see Figure 2). To test whether this difference was significant, we ran a crossed mixed-effects regression predicting total points earned on each trial with a fixed effect for the cost-modification condition (optimal vs. random) and random effects for both participant and problem. A likelihood ratio test of the mixed effects model with and without the condition fixed effect was significant ($\chi^2(1) = 37.711, p < 0.001$). Similar models predicting the click cost and decision quality also revealed significant effects, (click cost: 3.99 vs. 3.48, $\chi^2(1) = 8.6866, p = 0.003$, choice payout: 85.54 vs. 93.14, $\chi^2(1) = 33.821, p < 0.001$).

[1] Due to a programming error, the first test problem was always the same as the practice problem. We thus exclude data from this problem from our analysis, leaving 19 trials per participant.

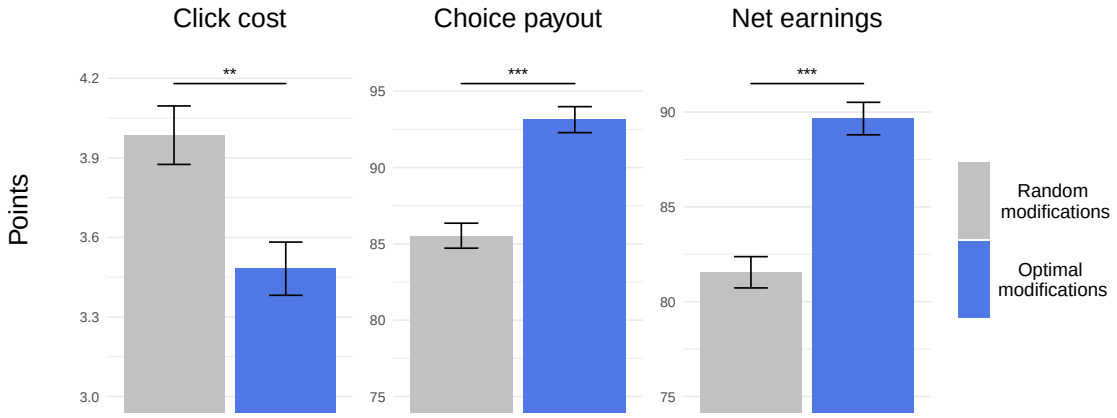


Figure 2: Average points per game spent and earned under random and optimal cost modifications. The first plot shows the average points spent uncovering values, the second the average reward from participants' choices, and the third their average net reward (choice payout minus click cost). Errors bars show standard error estimates derived from the residuals of the crossed mixed-effects regressions.

4 Discussion

In line with previous behavioral research on nudging, we find strong evidence that changes in choice architecture can have a significant impact on people's choices. Optimal modifications significantly increased participants' scores on Mouselab problems over random modifications, both increasing the average value of the options they chose and reducing the cost of making these choices. This provides preliminary evidence that modifying computational costs can be an effective way to help people make decisions more effectively.

Our approach has a number of advantages over other approaches to nudging. Not only can our framework be used to model existing nudges, we can use it motivate and develop new types of nudges, as we demonstrated by selecting modifications in order to maximize metalevel reward. However, nudges could be constructed with any number of different goals, for example, making people's decisions easier without systematically changing their choices. We provide a general and automatic way to construct nudges to optimize any objective. This objective could be selected by individuals, giving people control over how, when, and why they are nudged. We believe that this feature will be crucial in addressing ethical concerns with nudging.

One limitation of our approach is that it requires a fairly detailed model of the computational processes underlying the decision we would like to intervene on. In the present work, we avoided the challenge of developing such a model by using a process tracing paradigm that externalizes these typically unobservable processes. Applying the method in the real world, however, necessitates inferring such a model from observed behavior. This is a very difficult task. Nevertheless, even a heavily simplified model of the decision making process may be adequate to inform the construction of helpful, if not truly optimal, nudges.

References

Benartzi, S., Beshears, J., Milkman, K. L., Sunstein, C. R., Thaler, R. H., Shankar, M., … Galing, S. (2017). Should governments invest more in nudging? *Psychological Science*, *28*(8), 1041-1055. (PMID: 28581899)

Callaway, F., Lieder, F., Das, P., Gul, S., Krueger, P. M., & Griffiths, T. L. (2018). A resource-rational analysis of human planning. In *Proceedings of the 40th annual conference of the cognitive science society.*

Goodwin, T. (2012). Why we should reject "Nudge". *Politics*, *32*(2), 85-92.

Griffiths, T. L., Lieder, F., & Goodman, N. D. (2015). Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science*, *7*(2), 217-229.

Gul, S., Krueger, P. M., Callaway, F., Griffiths, T. L., & Lieder, F. (2018). Discovering rational heuristics for risky choice. In *The 14th biannual conference of the german society for cognitive science.*

Hay, N., Russell, S., Tolpin, D., & Shimony, S. E. (2012). Selecting computations: Theory and applications. In *Proceedings of the 28th conference on uncertainty in artificial intelligence.*

Kahneman, D., Slovic, P., & Tversky, A. e. (1982). *Judgment under uncertainty: heuristics and biases.* Cambridge, England: Cambridge University Press.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1988). Adaptive strategy selection in decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *14*(3), 534–552.

Russell, S., & Wefald, E. (1991). Principles of metareasoning. *Artificial Intelligence*, *49*(1-3), 361–395.

Thaler, R. H., & Sunstein, C. R. (2008). *Nudge: Improving decisions about health, wealth, and happiness.* New Haven: Yale University Press.

Robust Exploration with Tight Bayesian Plausibility Sets

Reazul H. Russel
Department of Computer Science
University of New Hampshire
Durham, NH 03824
rrussel@cs.unh.edu

Tianyi Gu
Department of Computer Science
University of New Hampshire
Durham, NH 03824
gu@cs.unh.edu

Marek Petrik
Department of Computer Science
University of New Hampshire
Durham, NH 03824
mpetrik@cs.unh.edu

Abstract

Optimism about the poorly understood states and actions is the main driving force of exploration for many provably-efficient reinforcement learning algorithms. We propose optimism in the face of sensible value functions (OFVF)- a novel *data-driven* Bayesian algorithm to constructing *Plausibility* sets for MDPs to explore robustly minimizing the worst case exploration cost. The method computes policies with tighter optimistic estimates for exploration by introducing two new ideas. First, it is based on Bayesian posterior distributions rather than distribution-free bounds. Second, OFVF does not construct plausibility sets as simple confidence intervals. Confidence intervals as plausibility sets are a sufficient but not a necessary condition. OFVF uses the structure of the value function to optimize the *location* and *shape* of the plausibility set to guarantee upper bounds directly without necessarily enforcing the requirement for the set to be a confidence interval. OFVF proceeds in an episodic manner, where the duration of the episode is fixed and known. Our algorithm is inherently Bayesian and can leverage prior information. Our theoretical analysis shows the robustness of OFVF, and the empirical results demonstrate its practical promise.

Keywords: Reinforcement Learning, Markov Decision Process, Exploration in RL, Bayesian Learning, Multi-armed bandits.

Acknowledgements

This work was supported by the National Science Foundation under Grant No. IIS-1717368 and IIS-1815275.

1 Introduction

Markov decision processes (MDPs) provide a versatile methodology for modeling dynamic decision problems under uncertainty [Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998; Puterman, 2005]. A perfect MDP model for many reinforcement learning problems is not known precisely in general. Instead, a reinforcement learning agent tries to maximize its cumulative payoff by interacting in an unknown environment with an effort to learn the underlying MDP model. It is important for the agent to explore sub-optimal actions to accelerate the MDP learning task which can help to optimize long-term performance. But it is also important to pick actions with highest known rewards to maximize short-run performance. So the agent always needs to balance between them to boost the performance of a learning algorithm during learning.

Optimism in the face of uncertainty (OFU) is a common principle for most reinforcement learning algorithms encouraging exploration [Auer *et al.*, 2010; Brafman and Tennenholtz, 2001; Kearns and Singh, 1998]. The idea is to assign a very high exploration bonus to poorly understood states and actions. As the agent visits and gathers statistically significant evidence for these states-actions, the uncertainty and optimism decreases converging to reality. Many RL algorithms including *Explicit Explore or Exploit* (E^3) [Kearns and Singh, 1998], *R-MAX* Brafman and Tennenholtz [2001], *UCRL2* [Auer, 2006; Auer *et al.*, 2010], *MBIE* [Strehl and Littman, 2008, 2004b,a; Wiering and Schmidhuber, 1998] build on the idea of optimism guiding the exploration. Probability matching class of algorithms like *Posterior Sampling for reinforcement learning (PSRL)* [Osband and Van Roy, 2017; Osband *et al.*, 2013; Strens, 2000] performs exploration with a proportional likelihood to the underlying true parameters. PSRL algorithm is simple, computationally efficient and can utilize any prior structural information to improve exploration. These algorithms provide strong theoretical guarantees with polynomial bound on sample complexity.

During exploration, it is possible for an agent to be overly optimistic about a potentially catastrophic situation and end up there paying an extremely high price (e.g. a self driving car hits a wall, a robot falls off the cliff etc.). Exploring and learning such a situation may not payoff the price. It can be wise for the agent to be robust and avoid those situations minimizing the worst-case exploration cost− which we call robust exploration. OFU and PSRL algorithms are optimistic by definition and cannot guarantee robustness while exploring. The main contribution of this paper is OFVF, an optimistic counter part of RSVF [Russel and Petrik, 2018]. OFVF is a Bayesian approach of constructing plausibility sets for robust exploration.

The paper is organized as follows: Section 2 formally defines the problem setup and goals of the paper. Section 3 reviews some existing methods to construct the plausibility sets and their extension to Bayesian setting. OFVF is proposed and analyzed in Section 4. Finally, Section 5 presents empirical performance on several problem domains.

2 Problem Statement

We consider the problem of learning a finite horizon Markov Decision Process $\mathscr{M}$ with states $\mathscr{S} = \{1, \ldots, S\}$ and actions $\mathscr{A} = \{1, \ldots, A\}$. $p : \mathscr{S} \times \mathscr{A} \to \Delta^{\mathscr{S}}$ is a transition function, where $p^a_{ss'}$ is interpreted as the probability of ending in state $s' \in \mathscr{S}$ by taking an action $a \in \mathscr{A}$ from state $s \in \mathscr{S}$. We omit s' when the next state is not deterministic and denote the transition probability as $p_{sa} \in \mathbb{R}^S$. $R : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ is a reward function and $R^a_{ss'}$ is the reward for taking action $a \in \mathscr{A}$ from state $s \in \mathscr{S}$ and reaching state $s' \in \mathscr{S}$. Each MDP $\mathscr{M}$ is associated with a discount factor $0 \leqslant \gamma \leqslant 1$ and a distribution of initial state probabilities p_0. We consider an episodic learning process where L is the number of episodes and H is the number of periods in each episode. A policy $\pi = (\pi_0, \ldots, \pi_{H-1})$ is a set of functions mapping a state $s \in \mathscr{S}$ to an action $a \in \mathscr{A}$. We define a value function for a policy π as:

$$V^\pi_h(s) := \sum_{s'} P^{\pi(s)}_{ss'}[r_h + V(s')] \tag{1}$$

The optimal value function is defined by $V^\star_h(s) = \max_\pi V^\pi_h(s)$ and the optimal policy is defined by $\pi^\star(s) = \arg\max_{a \in \mathscr{A}} p^a_{ss'} V(s'), \forall s' \in \mathscr{S} : p^a_{ss'} > 0$.

Optimistic algorithms encouraging exploration find the probability distribution $\tilde{P}_{sa}$ for each state and action within an interval of the empirically derived distribution $\bar{p}_{sa} = \mathbb{E}[\cdot|s, a]$, which defines the plausible set $\mathscr{P}_{sa}$ of MDPs. They then solve an optimistic version of Eq. (1) within $\mathscr{P}_{sa}$ that leads to the policy with highest reward.

$$V^\star_h(s, a) := \max_{p_{sa} \in \mathscr{P}_{sa}} \sum_{s'} p^{\pi(s)}_{ss'}[r_h + V^\star(s')] \tag{2}$$

We evaluate the performance of the agent in terms of worst-case *cumulative regret*, which is the maximum total regret incurred by the agent upto time T for a policy $\pi^\star_l$:

$$Regret(T, \pi^\star_l) = \sum_{l=0}^{T/H-1} \sup \Big[\sum_{s \in \mathscr{S}} p_0(s) \big(V^\star(s) - V^{\pi^\star_l}(s) \big) \Big] \tag{3}$$

Where $V^\star(s)$ is the true value w.r.t $\mathscr{M}*$.

3 Interval Estimation for Plausibility Sets

In this section, we first describe the standard approach to constructing plausibility sets as distribution free confidence intervals. We then propose its extension to Bayesian setting and present a simple algorithm to serve that purpose. It is important to note that distribution-free bounds are subtly different from the Bayesian bounds, the Bayesian safety guarantee holds conditional on a given dataset $\mathscr{D}$ while the distribution-free hold across the sets. This makes the guarantees qualitatively different and difficult to compare.

3.1 Plausibility Sets as Confidence Intervals

It is common in the literature to use L_1 norm as the distribution-free bound. This bound is constructed around the empirical mean of the transition probability $\bar{p}_{s,a}$ by applying the Hoeffding inequality [Auer *et al.*, 2010; Petrik *et al.*, 2016; Wiesemann *et al.*, 2013; Strehl and Littman, 2004b].

$$\mathscr{P}_{sa} = \left\{ \|\tilde{p}_{sa} - \bar{p}_{sa}\|_1 \leqslant \sqrt{\frac{2}{n_{s,a}} \log \frac{SA2^S}{\delta}} \right\}$$

where $\bar{p}_{sa}$ is the mean transition computed from D, $n_{s,a}$ is the number of times the agent arrived in state s' after taking action a in state s, δ is the required probability of the interval and $\|\cdot\|_1$ is the L_1 norm. An important limitation of this approach is that, the size of $\mathscr{P}_{sa}$ grows linearly with the number of states, which makes it practically useless in general.

3.2 Bayesian Plausibility Sets

The Bayesian plausibility sets take the same interval estimation idea and extend it into Bayesian setting, which is analogous to *credible intervals* in Bayesian statistics. Credible intervals are constructed with the posterior probability distributions and they are fixed − not a random variable, given the data $\mathscr{D}$. Instead the estimated transition probabilities maximizing the rewards are random variables. To construct a plausibility set, we optimize for the smallest credible region around the mean transition probability with the assumption that a smaller region will lead to a tighter upper bound estimate. Formally, the optimization problem to compute $\psi_{s,a}$ for each state s and action a is:

$$\min_{\psi \in \mathbb{R}_+} \{\psi \ : \ \mathbb{P}\left[\|\tilde{p}_{s,a} - \bar{p}_{s,a}\|_1 > \psi \mid \mathscr{D}\right] < \delta\} \ , \tag{4}$$

where nominal point is $\bar{p}_{s,a} = \mathbb{E}_{\tilde{P}}[\tilde{p}_{s,a} \mid \mathscr{D}]$. A Bayesian extension of the celebrated *UCRL* [Auer *et al.*, 2010] algorithm is *BayesUCRL*, which we consider for comparison. BayesUCRL algorithm uses a hierarchical Bayesian model that can be used to infer the posterior transition probability over $p^\star$. The plausibility set here is a function of the $\frac{1}{t}$-quantile of the posterior samples. We omit the details of BayesUCRL to conserve space.

4 OFVF: Optimism in the Face of sensible Value Functions

Algorithm 1: OFVF

Input: Desired confidence level δ and posterior distribution $\mathbb{P}_{P^\star}[\cdot \mid \mathscr{D}]$
Output: Policy with a maximized safe return estimate

Initialize current policy $\pi_0 \leftarrow \arg\max_\pi \rho(\pi, \mathbb{E}_{P^\star}[P^\star \mid \mathscr{D}])$;
Initialize current value $v_0 \leftarrow v^{\pi_0}_{\mathbb{E}_{P^\star}[P^\star \mid \mathscr{D}]}$;
Initialize value set $\mathscr{V}_0 \leftarrow \{v_0\}$;
Construct $\mathscr{P}_0$ optimal for $\mathscr{V}_0$;
Initialize counter $k \leftarrow 0$;
while *Eq.* (5) *is violated with* $\mathscr{V} = \{v_k\}$ **do**
    Include v_k that violates Eq. (5): $\mathscr{V}_{k+1} \leftarrow \mathscr{V}_k \cup \{v_k\}$;
    Construct $\mathscr{P}_{k+1}$ optimized for $\mathscr{V}_{k+1}$;
    Compute optimistic value function v_{k+1} and policy π_{k+1} for $\mathscr{P}_{k+1}$;
    $k \leftarrow k + 1$;
return $(\pi_k, p_0^\mathsf{T} v_k)$;

OFVF uses samples from a posterior distribution, similar to a Bayesian confidence interval, but it relaxes the safety requirement as it is sufficient to guarantee for each state s and action a that:

$$\min_{v \in \mathscr{V}} \mathbb{P}_{P^\star}\left[\max_{p \in \mathscr{P}_{s,a}} (p - p^\star_{s,a})^\mathsf{T} v \leqslant 0 \,\middle|\, \mathscr{D} \right] \geqslant 1 - \frac{\delta}{SA} \ , \tag{5}$$

with $\mathscr{V} = \{\hat{v}^\star_{\mathscr{P}}\}$. To construct the set $\mathscr{P}$ here, the set $\mathscr{V}$ is not fixed but depends on the optimistic solution, which in turn depends on $\mathscr{P}$. OFVF starts with a guess of a small set for $\mathscr{V}$ and then grows it, each time with the current value function, until it contains $\hat{v}^\star_{\mathscr{P}}$ which is always recomputed after constructing the ambiguity set $\mathscr{P}$.

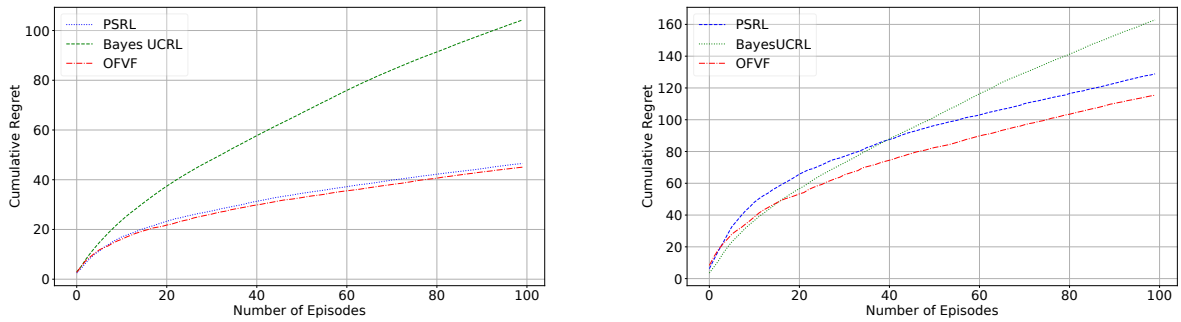


Figure 1: Cumulative regret for the single-state simple problem. Left) average-case, Right) worst-case.

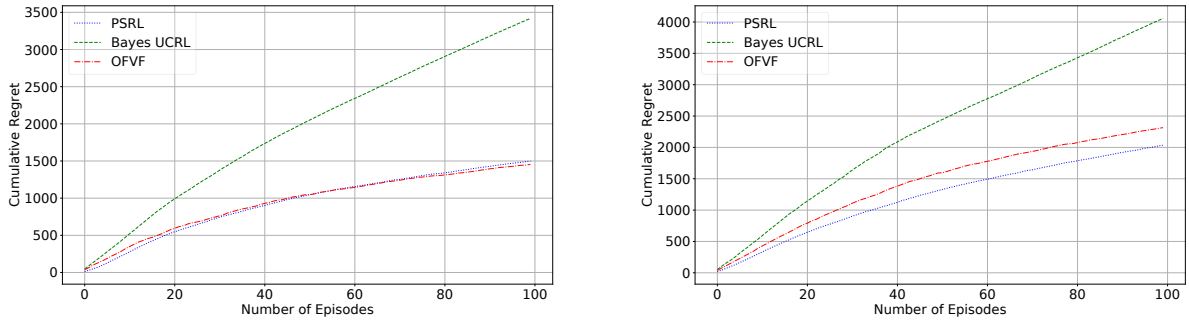


Figure 2: Cumulative regret for the RiverSwim problem. Left) average-case, Right) worst-case.

In lines 4 and 8 of Algorithm 1, $\mathscr{P}_i$ is computed for each state-action $s, a \in \mathscr{S} \times \mathscr{A}$. Center $\bar{p}$ and set size $\psi_{s,a}$ are computed from Eq. (7) using set $\mathscr{V}$ & optimal g_v computed by solving Eq. (6). When the set $\mathscr{V}$ is a singleton, it is easy to compute a form of an optimal plausibility set.

$$g = \max\left\{k \;:\; \mathbb{P}_{P^\star}[k \leqslant v^\mathsf{T} p^\star_{s,a}] \geqslant 1 - \delta/(SA)\right\} \tag{6}$$

For a singleton $\mathscr{V}$, it is sufficient for the plausibility set to be a subset of the hyperplane $\{p \in \Delta^S \;:\; v^\mathsf{T} p = g^\star\}$ for the estimate to be sufficiently optimistic. When $\mathscr{V}$ is not a singleton, we only consider the setting when it is discrete, finite, and relatively small. We propose to construct a set defined in terms of an L_1 ball with the minimum radius such that it is safe for every $v \in \mathscr{V}$. Assuming that $\mathscr{V} = \{v_1, v_2, \ldots, v_k\}$, we solve the following linear program:

$$\psi_{s,a} = \min_{p \in \Delta^S}\left\{\max_{i=1,\ldots,k} \|q_i - p\|_1 \;:\; v_i^\mathsf{T} q_i = g_i^\star, q_i \in \Delta^S, i \in 1, \ldots, k\right\} \tag{7}$$

In other words, we construct the set to minimize its radius while still intersecting the hyperplane for each v in $\mathscr{V}$.

5 Empirical Evaluation

In this section, we empirically evaluate the estimated returns over episodes. We assume a true model of each problem and generate a number of simulated data sets for the known distribution. We compute the tightest optimistic estimate for the optimal return and compare it with the optimal return for the true model. To judge the performance of the methods, we evaluate both the absolute error of the worst case estimates from optimal, as well the average case estimate from optimal.

We compare our results with BayesUCRL and PSRL algorithms. We omit UCRL from comparison because it performs too poorly compared to other methods. PSRL performs very well in both average and worst case, and as we will see in the experiments, OFVF outperforms BayesUCRL and performs competitively with PSRL. For all the experiments, we use an uninformative Dirichlet prior for the transition probabilities, and run experiments for 100 episodes each containing 100 runs, unless otherwise specified.

Single-state Bellman Update We initially consider a simple problem with one single non-terminal state. The agent can take three different actions on that state. Each action leads to one of three terminal states with different transition probabilities. The value function for the terminal states are fixed and assumed to be known. Fig. 1 compares the average-case and worst-case returns computed by different methods. Note that OFVF outperforms all other methods in this simplistic setting. OFVF is able to explore in a robust way maximizing the worst and average case returns.

RiverSwim Problem We compare the performance of different methods in standard example of RiverSwim [Osband *et al.*, 2013; Strehl and Littman, 2004b]. The problem is designed requiring hard exploration to find the optimal policy, we omit the full description of the problem to preserve space. Fig. 2 compares the average and worst case regrets of different methods. Among optimistic methods, OFVF performs better than BayesUCRL both in average and worst case scenario. But the stochastically optimistic PSRL outperforms all other methods. This is due to the fact that, BayesUCRL and OFVF constructs a plausibility set for each state and action. Even if the plausibility sets are tight, the resulting optimistic MDP is simultaneously optimistic in each state-action, yielding a way too optimistic overall MDP model [Osband and Van Roy, 2017]. Thus OFVF can construct tighter plausibility sets for exploration, but still may not match the statistical efficiency of PSRL. This performance however shows that, as an OFU algorithm, OFVF can be reasonably optimistic and can offer competitive performance.

6 Summary and Conclusion

In this paper, we proposed OFVF, a Bayesian algorithm capable of constructing plausibility sets with better shapes and sizes. Beside the fact that our proposed Bayesian methods are computationally demanding than other distribution free methods, our theoretical and experimental analysis furnished that they can pay-off with much tighter return estimates. We showed that, OFU algorithms can be useful and can be competitive to stochastically optimistic algorithm like PSRL.

References

P Auer, Thomas Jaksch, and R Ortner. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(1):1563–1600, 2010.

Peter Auer. Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning. *Advances in Neural Information Processing Systems (NIPS)*, 2006.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *International Conference on Machine Learning (ICML)*, 1998.

Ian Osband and Benjamin Van Roy. Why is Posterior Sampling Better than Optimism for Reinforcement Learning? *International Conference on Machine Learning (ICML)*, 2017.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) Efficient Reinforcement Learning via Posterior Sampling? *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Marek Petrik, Yinlam Chow, and Mohammad Ghavamzadeh. Safe Policy Improvement by Minimizing Robust Baseline Regret. *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 2005.

Reazul Hasan Russel and Marek Petrik. Tight Bayesian Ambiguity Sets for Robust MDPs. *Infer to Control, Workshop on Probabilistic Reinforcement Learning and Structured Control, Advances in Neural Information Processing Systems (NIPS)*, 2018.

Alexander. L. Strehl and Michael L. Littman. An empirical evaluation of interval estimation for markov decision processes. *IEEE International Conference on Tools with Artificial Intelligence*, 2004.

Alexander L Strehl and Michael L Littman. Exploration via Model-based Interval Estimation. *International Conference on Machine Learning (ICML)*, 2004.

Alexander L Strehl and Michael L Littman. An Analysis of Model-Based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 74:1309–1331, 2008.

Malcolm Strens. A Bayesian Framework for Reinforcement Learning. *International Conference on Machine Learning (ICML)*, 2000.

Richard S Sutton and Andrew Barto. Reinforcement Learning: An Introduction. 1998.

Marco Wiering and Jurgen Schmidhuber. Efficient Model-Based Exploration. *International Conference on Simulation of Adaptive Behavior (SAB)*, pages 223–228, 1998.

Wolfram Wiesemann, Daniel Kuhn, and Berc Rustem. Robust Markov Decision Processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

On Inductive Biases in Deep Reinforcement Learning

Matteo Hessel *
DeepMind
London, UK
mtthss@google.com

Hado van Hasselt *
DeepMind
London, UK
hado@google.com

Joseph Modayil
DeepMind
London, UK
modayil@google.com

David Silver
DeepMind
London, UK
davidsilver@google.com

Abstract

Many deep reinforcement learning algorithms contain inductive biases that sculpt the agent's objective and its interface to the environment. These inductive biases can take many forms, including domain knowledge and pretuned hyper-parameters. In general, there is a trade-off between generality and performance when we use such biases. Stronger biases can lead to faster learning, but weaker biases can potentially lead to more general algorithms that work on a wider class of problems. This trade-off is relevant because these inductive biases are not free; substantial effort may be required to obtain relevant domain knowledge or to tune hyper-parameters effectively. In this paper, we re-examine several domain-specific components that bias the objective and the environmental interface of common deep reinforcement learning agents. We investigated whether the performance deteriorates when these components are replaced with adaptive solutions from the literature. In our experiments, performance sometimes decreased with the adaptive components, as one might expect when comparing to components crafted for the domain, but sometimes the adaptive components performed better. We then investigated the main benefit of having fewer domain-specific components, by comparing the learning performance of the two systems on a different set of continuous control problems, without additional tuning of either system. As hypothesized, the system with adaptive components performed better on many of the new tasks.

Keywords: Reinforcement learning, deep learning, inductive biases

*equal contribution

1 Introduction

The deep reinforcement learning (RL) community has demonstrated that well-tuned deep RL algorithms can master a wide range of tasks, including board games [1], video-games [2], and custom 3D navigation tasks [3]. These results are a testament to the generality of the approach. At times, however, the excitement for the stream of challenges being mastered by RL agents may have over-shadowed the dependency of some of these agents on various forms of inductive biases, and the amount of tuning required for them to perform well.

In general, there is a trade off between generality and performance when we inject inductive biases into our algorithms. Inductive biases take many forms, including domain knowledge and pretuned learning parameters. If applied carefully, such biases can lead to faster and better learning. On the other hand, fewer biases can potentially lead to more general algorithms that work out of the box on a wider class of problems. A clear example of the benefits of generality is the AlphaZero algorithm [1], which, by removing all Go-specific inductive biases of the predecessor AlphaGo, could not just surpass human performance on Go but also learn to play Chess and Shogi. Importantly, inductive biases are typically not free: substantial effort is required to attain the relevant domain knowledge or pretune parameters. This cost is often hidden—e.g., one may use hyperparameters enstablished as good in prior work on the same domain, without knowing how much time was spent optimising them, nor how specific they are to the domain. But this cost is a huge bottleneck when applying even well known algorithms to new domains. Systematic studies about the impact of the various inductive biases are rare and, as a result, the generality of common biases in deep RL is often unclear.

We consider two broad ways of injecting inductive biases in RL agents: 1) sculpting the agent's objective (e.g., clipping and discounting rewards); 2) sculpting the agent-environment interface (e.g., using fixed action repetitions, or crafting the observations). We then investigate if (and by how much) performance deteriorates when replaced with general adaptive components. For instance, we show that all the carefully crafted heuristics commonly used in Atari, and often considered essential for good performance on this domain, can be replaced with adaptive components, while preserving competitive performance across the benchmark. Furthermore, we show that this results in increased generality for an actor critic agent; the resulting fully adaptive system can be applied with no additional tuning on a separate suite of continuous control tasks, with much higher performance than a comparable system using Atari tuned heuristics, and even higher performance than an actor-critic agent tuned for this benchmark specifically by [4].

2 Background

Problem setting: Reinforcement learning is a framework for learning and decision making under uncertainty, where an *agent* interacts with its *environment*, by executing actions A_t and receiving *observations* O_{t+1} and *rewards* R_{t+1} in return. The behaviour of an agent is specified by a *policy* $\pi(A_t|H_t)$: a probability distribution over actions conditional on all previous observations, i.e. on the *history* $H_t = O_{1:t}$. The agent's objective is to find a policy that collects as much reward as possible, in each episode of experience. Crucially, it must learn such a policy without direct supervision, by trial and error. The amount of reward collected from time t onwards - the *return* - is a random variable

$$G_t = \sum_{k=0}^{T_{end}} \gamma^k R_{t+k+1}, \tag{1}$$

where T_{end} is the number of steps until episode termination and $\gamma \in [0, 1]$ is a constant discount factor. The agent seeks an *optimal* policy, that maximizes *values* $v(H_t) = E_\pi[G_t|H_t]$. In *fully observable* environments the optimal policy depends on the last observation alone: $\pi^*(A_t|H_t) = \pi^*(A_t|O_t)$. Otherwise, the history may be summarized in an *agent state* $S_t = f(H_t)$. The agent's objective is then to *jointly* learn the state representation f and policy $\pi(A_t|S_t)$, so as to maximize values. The fully observable case is formalized as a Markov Decision Process [5].

Actor-critic algorithms: Value-based algorithms efficiently learn to approximate values $v_\mathbf{w}(s) \approx v_\pi(s) \equiv E_\pi[G_t|S_t = s]$, under a policy π, by exploiting a recursive decomposition $v_\pi(s) = E[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$, known as Bellman equation, as done in temporal difference learning [6] through sampling and incremental updates:

$$\Delta\mathbf{w}_t = (R_{t+1} + \gamma v_\mathbf{w}(S_{t+1}) - v_\mathbf{w}(S_t))\nabla_\mathbf{w} v_\mathbf{w}(S_t). \tag{2}$$

Policy-based algorithms update directly a parameterized policy $\pi_{\boldsymbol{\theta}}(A_t|S_t)$ through a stochastic gradient estimate of the direction of steepest ascent in the value [7, 8]; for instance, according to the update

$$\Delta\boldsymbol{\theta}_t = G_t \nabla \log \pi_{\boldsymbol{\theta}}(A_t|S_t). \tag{3}$$

Value-based and policy-based methods are combined in *actor-critic* algorithms. If a state value estimate is available, the policy updates can be computed from incomplete episodes by using the truncated returns $G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n v_\mathbf{w}(S_t)$ that bootstrap on the value estimate at state S_{t+n} according to $v_\mathbf{w}$. This can reduce the variance of the updates. The variance can be further reduced using state values as a baseline in policy updates, as in *advantage* actor-critic updates

$$\Delta\boldsymbol{\theta}_t = (G_t^{(n)} - v_\mathbf{w}(S_t))\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t). \tag{4}$$

3 Inductive Biases and Learned Solutions

Sculpting the agent's objective: agents typically do not directly optimize the objective that they are evaluated against. They optimize a different handcrafted objective incorporating biases to simplify learning. We consider 2 ways of sculpting such objective: reward clipping, and fixed reward discounting by a factor different from the evaluation discount.

In many deep RL algorithms, the magnitude of updates scales linearly with the returns. This complicates training the same RL agent, with same hyper-parameters, on multiple domains, because good settings for hyper-parameters vary across tasks. A common solution is to clip rewards to a fixed range [9], for instance $[-1, 1]$. This makes the magnitude of returns and updates more comparable. However, it also changes the agent objective. This can make learning simpler, and, when it is a good proxy for the true objective, results in good performance. In some tasks, however, clipping results in sub-optimal policies. PopArt [10, 11] was introduced to learn effectively irrespective of scale. PopArt works by computing temporal difference errors used for updates 2 and 4 in a normalized space. This is done by reparametrizing values as the linear transformation $v_{\mathbf{w}}(s) = \mu + \sigma * n_{\mathbf{w}}(s)$ of normalized values $n_{\mathbf{w}}(s)$, where μ and σ are computed by tracking *mean* and *standard deviation* of the bootstrapped returns. PopArt additionally combines such adaptive rescaling of values with an inverse transformation of the weights at the last layer of $n_{\mathbf{w}}(s)$, to preserve outputs precisely under any change in statistics $\mu \to \mu'$ and $\sigma \to \sigma'$. Note that this can be done *exactly*.

Discounting is part of the traditional MDP formulation of RL. As such, it is often considered a property of the problem rather than a parameter of the agent. Indeed, sometimes, the environment does define a *natural* discounting of rewards (e.g., inflation in a financial setting). However, even in settings where the agent *should* maximize the undiscounted return, a constant discount is often used to simplify the problem, as optimizing such proxy objective often results in superior performance even in terms of undiscounted return. This, however, comes at the cost of adding a hyperparameter, and a rather sensitive one—i.e., learning may be fast if the discount is small, but the solution may be myopic. Instead of *tuning* the discount, we use meta-learning [12] to adapt it. The meta-gradient algorithm by [13] exploits the fact that updates such as 2 and 4 are differentiable functions of the discount. On a second sample of experience, generated with updated parameters $\mathbf{w} + \Delta\mathbf{w}(\gamma)$, the agent can therefore apply an actor-critic update, not to $\mathbf{w}$ but to the discount γ used to update $\mathbf{w}$. [13] demonstrated that this improved performance, while using a separate hand-tuned discount factor for the meta-update. We use the undiscounted returns ($\gamma_m = 1$) to compute the meta-gradients, to understand if this technique can fully replace the need to reason about timescales.

Sculpting the agent-environment interface: In RL we assume that time progresses in discrete steps with a fixed duration. While algorithms are typically defined in this space, learning at the fastest timescale provided by the environment is often not practical. It is often convenient to have the agent operate at a slower timescale, for instance by repeating each selected action a fixed number of times. The use of such fixed action repetitions is a widely used heuristic [9] with several advantages. 1) Operating at a slower timescale increases action gaps, which can lead to more stable learning; 2) selecting an action every few steps can save a significant amount of computation; 3) committing longer to each action may help exploration, e.g., by removing often-irrelevant sequences of actions that jitter back and forth. A more general solution approach is for the agent to *learn* the most appropriate time scale at which to operate. Solving this problem in full generality is one of the aims of hierarchical reinforcement learning [14]. This general problem remains largely unsolved. A simpler, though more limited, approach is to allow the agent to learn how long to commit to each selected action [15]. At each step t, the agent may be allowed to select both an action A_t and a commitment C_t, by sampling from two separate policies, both trained with policy gradient.

Many state-of-the-art RL agents use non-linear function approximators to represent values, policies, and states. Being able to learn flexible state representations was essential to capitalize on the successes of deep learning, and to scale up reinforcement learning algorithms. While the use of deep neural network to approximate value functions and policies is widespread, their input is often not the raw observations but the result of domain-specific heuristic transformations. In Atari, for instance, most agents rely on down-sampling observations to a coarser resolution, max-pooling consecutive frames to remove flickering, grey scaling, and finally concatenating of frames into a *K-Markov* representation. We replace this specific preprocessing pipeline with a recurrent state representation learned end-to-end.

4 Experiments

When designing algorithms it is useful to keep in mind what properties we would like the algorithm to satisfy. If the aim is to design an algorithm, or inductive bias, that is *general*, in addition to metrics such as asymptotic performance and data efficiency, there are additional dimensions that are useful to consider. 1) Does the algorithm require careful reasoning to select an appropriate time horizon for decision making? This is tricky without domain knowledge or tuning. 2) How robust is the algorithm to the scaling of rewards? Rewards can have arbitrary scales, that may change by orders of magnitudes during training. 3) Can the agent use commitment (e.g. action repetitions, or options) to alleviate the difficulty of learning at the fastest time scale? 4) Does the algorithm scale effectively to large complex problems? 5) Does the algorithm generalize well to problems it was not specifically designed and tuned for?

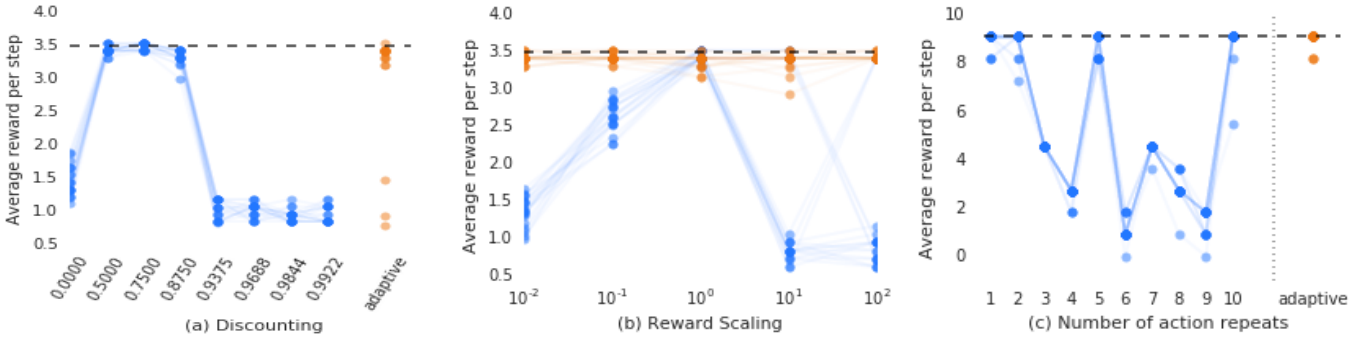


Figure 1: Investigations on the robustness of an A2C agent with respect to discounting, reward scaling and action repetitions. We report the average reward per environment step, after 5000 steps of training, for each of 20 distinct seeds. Each parameter study compares different fixed configurations of a specific hyper-parameter to the corresponding adaptive solution. In all cases the performance of the adaptive solutions is competitive with that of the best tuned solution

We first investigate how current domain heuristics and adaptive solutions compare with respect to dimensions 1-3, by training a number of tabular actor-critic agent in a few small chain environments. Figure 1a shows a parameter study over a range of values for the discount factor (in blue) in the first of these environments. It highlights how it can be difficult to set a suitable discount factor, and that naively optimizing undiscounted returns can also perform poorly. Compare this to the same agent, but equipped with the meta-gradient algorithm (in orange in Figure 1.a). Even initializing the discount adversarially, the agent learned to adapt the discount and performed in par with the best tuned fixed discount. Figure 1.b shows the impact of reward scaling by considering the performance of the tabular actor-critic agent when rewards are scaled by different factors. The performance of the naive actor-critic (in blue) is quite sensitive to the scale, while for the same agent equipped with PopArt we observed good performance across all tested scales. Finally, Figure 1.c compares an actor-critic agent that learns to choose the number of action repeats, to agents with different fixed repetitions C. This fixed number of action repetitions can also be a sensitive hyper-parameter, while the adaptive agent that learns how often to repeat actions via policy gradient (in orange in Figure 1.c) can easily learn a suitable number of action repetitions.

Next we investigate the properties of the various inductive biases and adaptive solutions at scale. We train several actor-critic agents with non-linear function approximation on each of 57 Atari games. We first measure the performance of a fully adaptive agent with learned action repeats, PopArt normalization, learned discount factors, and an LSTM-based state representation. We compare the performance of this agent to agents with exactly one adaptive component disabled and replaced with one of two fixed components. This fixed component corresponds to either falling back to the naive solution (e.g. learning directly from undiscounted returns), or using the corresponding domain heuristic. This enables us to investigate how important the heuristic is for current RL algorithms, as well as how fully current adaptive solution can replace it. Figure 2a shows that in the first 100M frames, the agent acting at the fastest rate is competitive with the agents equipped with action repetition (fixed or learned). However, while the agents with action repeats still improve performance until the very end of training, the agent acting at the fastest timescale plateaus much earlier. This performance plateau is observed across many games, and we speculate that the use of multiple action repetitions may be providing better exploration. Learning to repeat performed comparably to the tuned domain heuristic in Atari. Figure 2b show that using undiscounted returns directly in the updates to policy and values results in poor performance; this confirms our intuition that directly optimizing the real objective is problematic with current deep RL solutions, and the large effect of biasing the objective towards simpler proxies. However, using the undiscounted objective for the meta-gradient updates and learning the discounting does work well in practice, and slightly outperforms the tuned heuristic. In Figure 2c shows that the naive solution of learning from the raw environment rewards performs very poorly, compared to using either reward clipping or the learned solution. Note however that here the domain heuristic (reward clipping) retained a significant edge over PopArt. This suggests that the inductive bias of optimizing for a weighted frequency of rewards is a very good heuristic in many Atari games. Finally, in Figure 2d we compare the fully end to end pipeline with a recurrent network, to a feedforward neural network with the standard Atari pipeline. The recurrent end to end solution performed best, showing that a recurrent network is sufficiently flexible to learn on its own to integrate relevant information over time, despite Atari-specific issues such as the flickering of the screen. Finally, in order to investigated the generality of these different RL solutions, we compared the fully general agent to an agent with all the usual inductive biases, but in the context of a completely different benchmark: a collection of 28 continuous control tasks. For both we use the exact same solutions that were used in Atari, with no additional tuning. Figure 2e shows that the fully general agent performed better than the heuristic solution, which suggests that the set of inductive biases typically used by Deep RL agents on Atari do not generalize as well as the set of adaptive solutions considered in this paper. The adaptive solution was also better, overall, than the tuned baseline by [4], as shown by the reference horizontal line.

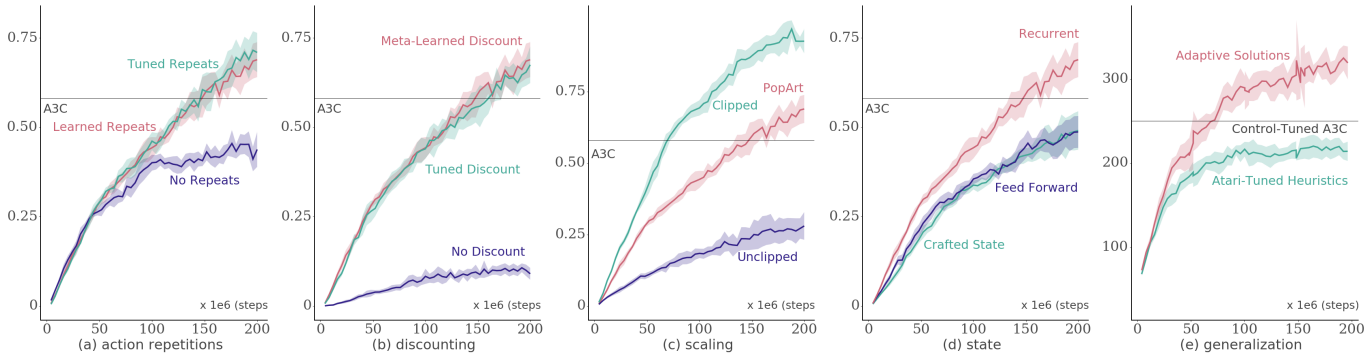


Figure 2: Comparison of inductive biases to RL solutions. The first four plots compares median performance across 57 Atari games for the same fully general agent and 2 alternatives: a) tuned action repeats, and no action repeats. b) tuned discount factor, and no discounting. c) reward clipping, and learning from raw rewards. d) learning from raw observations, and the standard preprocessing. The right-most plot (e) compares mean performance across 28 continuous control tasks for the fully adaptive actor critic agent and an agent the same biases as in the Atari experiments.

5 Conclusion

We found existing learned solutions are competitive with well tuned domain heuristics, even on the domain these heuristics were designed for, and they generalize better to unseen domain. This suggests removing these biases in future research, since they are not essential for performance, and they might hide issues in the learning algorithms. There are two features of our agent that, despite not incorporating quite as much domain knowledge as the heuristics discussed in the paper, may also affect its generality: 1) The use of parallel copies of the environment in actor-critic agents is not always practical, especially in real world applications. 2) Back-propagation through time for training RNNs constrains the length of the temporal relationship that we can learn. We believe further work on these issues to be important.

Our work was inspired by that of [1] in the context of Go, but we investigated the different set of domain specific heuristics, that are used in more traditional deep RL agents. Our work also relates to a broader debate [16] about priors and innateness. There is evidence that we, as humans, posses specific types of biases, and that these have a role in enabling efficient learning [17]; however, it is not clear whether these are *essential* for intelligent behaviour to arise, nor what form such priors take, and their generality. Here we show that several heuristics we use in deep RL are harming the generality of our methods. This does not imply that *different* inductive biases could not be useful, but it is a reminder that we must be careful with the domain knowledge we bake into algorithms, and we must revise these biases over time.

References

[1] Silver, Hubert, Schrittwieser, Antonoglou, Lai, Guez, Lanctot, Sifre, Kumaran, Graepel, Lillicrap, Simonyan, and Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017.

[2] Hessel, Modayil, van Hasselt, Schaul, Ostrovski, Dabney, Horgan, Piot, Azar, and Silver. Rainbow. *AAAI*, 2018.

[3] Kempka, Wydmuch, Runc, Toczek, and Jaśkowski. Vizdoom: A doom-based ai research platform for visual rl. In *CIG*, 2016.

[4] Tassa, Doron, Muldal, Erez, Li, Las Casas, Budden, Abdolmaleki, Merel, Lefrancq, Lillicrap, and Riedmiller. Control suite. 2018.

[5] Richard Bellman. *Dynamic programming*. Princeton University Press, 1957.

[6] Sutton. Learning to predict by the methods of temporal differences. *ML*, 1988.

[7] Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *ML*, 1992.

[8] Sutton, McAllester, Singh, and Mansour. Policy gradient methods for rl with function approximation. In *NIPS*, 2000.

[9] Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, and Kavukcuoglu. Asynchronous methods for deep rl. In *ICML*, 2016.

[10] van Hasselt, Guez, Hessel, Mnih, and Silver. Learning values across many orders of magnitude. In *NIPS*, 2016.

[11] Hessel, Soyer, Espeholt, Czarnecki, Schmitt, and van Hasselt. Multi-task deep reinforcement learning with popart. *AAAI*, 2018.

[12] Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, 1992.

[13] Xu, van Hasselt, and Silver. Meta-gradient reinforcement learning. *CoRR*, abs/1805.09801, 2018.

[14] Sutton., Precup, and Singh. Intra-option learning about temporally abstract actions. In *ICML*, 1998.

[15] Lakshminarayanan, Sharma, and Ravindran. Dynamic action repetition for deep reinforcement learning. In *AAAI*, 2017.

[16] Marcus. Innateness, alphazero, and artificial intelligence. *CoRR*, abs/1801.05667, 2018.

[17] Dubey, Agrawal, Pathak, Griffiths, and Efros. Investigating human priors for playing video games. *CoRR*, abs/1802.10217, 2018.

Symbolic Planning and Model-Free Reinforcement Learning: Training Taskable Agents

León Illanes
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
lillanes@cs.toronto.edu

Xi Yan
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
xi.yan@mail.utoronto.ca

Rodrigo Toro Icarte
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
rntoro@cs.toronto.edu

Sheila A. McIlraith
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada
sheila@cs.toronto.edu

Abstract

We investigate the use of explicit symbolic action models, as typically used for Automated Planning, in the context of Reinforcement Learning (RL). Our objective is to make RL agents more sample efficient and human taskable. We say an agent is taskable when it is capable of achieving a variety of different goals and there is a simple method for goal specification. Moreover, we expect taskable agents to easily transfer skills learned for one task to other related tasks. To these ends, we consider high-level models that inexactly represent the low-level environment in which an agent acts. Given a model, defining goal-directed tasks is a simple problem, and we show how to communicate these goals to an agent by leveraging state-of-the-art symbolic planning techniques. We automatically generate families of high-level solutions and subsequently represent them as a reward machine, a recently introduced formalism for describing structured reward functions. In doing this, we not only specify what the task at hand is, but also give a high-level description of how to achieve it. The structure present in this description can be successfully exploited by a Hierarchical RL system. The reward machine represents a collection of sequential solutions and can be used to prune the options available when training. We can ensure that, at every step, the meta-controller can only select options that represent advancement in some high-level plan.

We empirically demonstrate the merits of our approach, comparing to a naive baseline where a single sequential plan is strictly followed, and to standard Hierarchical RL techniques. Our results show that the approach is an effective method for specifying tasks to an RL agent. Given adequately pretrained options, our approach reaches high-quality policies in previously unseen tasks in extremely few training steps and consistently outperforms the standard techniques.

Keywords: Reinforcement Learning
Hierarchical Reinforcement Learning
Goal Specification
Automated Planning
Symbolic Planning

Acknowledgements

We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC), Becas Chile (CONICYT), and Microsoft Research.

1 Introduction

Reinforcement learning (RL) techniques allow for agents to perform tasks in complex domains, where environment dynamics and reward structures are initially unknown. These techniques are based on performing random exploration, observing the dynamics and reward returned by the environment, and synthesizing an optimal policy that maximizes expected cumulative reward. Unfortunately, when reward is sparsely distributed, as is the case in many applications, RL techniques can suffer from poor *sample efficiency*, requiring millions of episodes to learn reasonable policies. Further, these systems are typically not *taskable*: specifying new tasks is often difficult and the skills that are learned for one task are not easily transferred to others. Over the years a number of approaches have been proposed to address these shortcomings including efforts to learn hierarchical representations or to define *options*, a form of macro-action that can be used by the RL system [9] .

Our interest in this paper is in leveraging high-level symbolic planning models and automated plan synthesis techniques, in concert with state-of-the-art RL techniques, with the objective of significantly improving sample efficiency and creating systems that are human taskable. Our efforts are based on the observation that some approximated understanding of the environment can be characterized as a symbolic planning model—a set of properties of the world and actions that cause those properties to change in predictable ways.

Recent research has demonstrated the benefit of providing high-level instructions to an RL system to improve sample efficiency (e.g., policy sketches [1]). Nevertheless, these instructions must be *manually* generated. By using a symbolic model of the environment rather than a task-specific set of instructions we are able to *automatically* generate instructions in the form of one or more sequential or partial order plans—the latter compactly characterizing a multitude of sequential plans. We represent these plans as a structured reward function in the form of an automata-inspired reward machine [10]. This reward machine is then used to enhance a Hierarchical Reinforcement Learning (HRL) system by ignoring options that—based on inspection of the machine's structure—cannot result in future reward.

We compare our approach to standard forms of HRL and to a naive baseline algorithm. Our results show that the approach is an effective method for specifying tasks to an RL agent, reaching high-quality policies for previously unseen tasks in extremely few training steps.

2 Related Work

The options framework [9] has become a well-known standard approach for exploiting temporal abstraction in Reinforcement Learning. A key contribution of our work is the way in which we use explicit symbolic planning to select adequate options.

Other existing approaches have used symbolic planning to select options or macro-actions. An early approach proposed using a symbolic planner coupled into an RL agent [2]. There, the planner produces an initial high-level plan and is subsequently used to replan when the plan's preconditions are violated. A related approach uses a sequential plan to modify the reward via reward shaping [3]. In contrast to these approaches, our system uses planners to produce a single structured reward function that represents the task at hand. Our approach is orthogonal to the idea of reward shaping, and we believe that it may be applied in our setting.

Recent work has also proposed methods based on coupling a planner to an RL agent [11, 7]. There, the focus has been on two-way communication between agent and planner to continuously improve the high-level model and find better high-level solutions. In our case, we assume an adequate high-level model is given and we show how to exploit it. Other work has explored techniques for automatically building such abstractions [6, 5]. Integrating such techniques into our work may prove to be an interesting direction for future work.

Finally, there has also been work that has focused on learning explicit state-transition systems that represent high-level models [12]. With these, standard graph search algorithms can be used to find sequences of macro-actions. Our work considers *implicit* state-transition systems described as classical planning domains. This allows us to consider highly combinatorial problems that are far too large to represent explicitly.

3 Preliminaries

For the purposes of this work, we will say that the environment in which an RL agent acts is formalized as a tuple $\mathcal{E} = \langle S, A, p \rangle$, where S is its set of states, A is the set of available actions, and $p(s_{t+1} \mid s_t, a_t)$ is the transition probability distribution. A policy is defined as a probability distribution $\pi(a \mid s)$ that establishes the probability of the agent taking action a given that its current state is s.

For defining tasks, we consider a recently introduced reward mechanism known as reward machines [10]. A reward machine is a finite-state machine that can be used to specify temporally-extended and non-Markovian reward functions. The intuitive idea is that transitions in the reward machine take place based on observations made by the agent about the environment, and that the reward depends on the transitions taken in the machine. The observations are represented by a set of propositional symbols $\mathcal{P}$, which correspond to facts that the agent may perceive. For a given environment, we

assume there is a labeling function $L\colon S \to 2^{\mathcal{P}}$ that establishes what is perceived when reaching a state. Then, a reward machine for environment $\mathcal{E}$ and observation propositions $\mathcal{P}$ is given by the tuple $\mathcal{R} = \langle U, u_0, \delta_u, \delta_r \rangle$. U is its set of states, u_0 is its initial state, $\delta_u\colon U \times 2^{\mathcal{P}} \to U$ is its state transition function, and $\delta_r\colon U \times U \to \mathbb{R}$ is its reward transition function. Whenever the agent makes a transition (s, a, s') in the environment and observes $P \subseteq \mathcal{P}$, the current state in the reward machine is updated from u to $u' = \delta_u(u, P)$. At this point, the agent receives reward $\delta_r(u, u')$.

Temporal Abstraction in RL The options framework proposes the use of policies that are trained for achieving specific high-level behaviours, coupled with well-defined termination criteria for their application [9]. An agent acting within this framework can choose, at every step, to either apply a low-level action or to apply one of these high-level options. We use a simplistic notion of option defined as a pair $o = \langle \pi_o, T_o \rangle$, where π_o is the corresponding policy and $T_o \subseteq S$ is a set of states that the policy is intended to reach and that will be used as the option's termination criterion.

Symbolic Planning We specify planning domains in terms of a tuple $\mathcal{D} = \langle F, \mathcal{A} \rangle$. F is a set of propositional symbols, called the fluents of $\mathcal{D}$, and $\mathcal{A}$ is the set of planning actions in the domain. Planning states are specified as subsets of F, so that state $S \subseteq F$ represents the situation in which the fluents in S are all true and those not in S are false. Actions are specified in terms of their preconditions and effects, which are given as logical formulae over the propositional fluents. Actions are only applicable in states where their preconditions are satisfied, and such application results in a transition to a state where the action's effects are true, and everything else remains unchanged.

A planning task is described by an initial state and a goal condition. The goal is given as a formula over the fluents of the domain. Any state that satisfies the goal condition is said to be a goal state. A sequence of actions $\Pi = [a_0, a_1, \ldots, a_n]$ is known as a sequential plan for a task when it is possible to sequentially apply the actions starting at the initial state, and doing so reaches a goal state. Given a plan $\Pi = [a_0, a_1, \ldots, a_n]$, we will refer to its prefix with respect to action a_i as **prefix**$(\Pi, a_i) = [a_0, a_1, \ldots, a_{i-1}]$.

Partial-order plans generalize sequential plans by relaxing the ordering condition over the actions. A partial-order plan is a tuple $\overline{\Pi} = \langle \overline{\mathcal{A}}, \prec \rangle$, where $\overline{\mathcal{A}}$ is its set of action occurrences and $\prec$ is a partial order over $\overline{\mathcal{A}}$. The set of linearizations of $\overline{\Pi}$, denoted $\Lambda(\overline{\Pi})$, is the set of all sequences of the action occurrences in $\overline{\mathcal{A}}$ that respect the partial order $\prec$. Any linearization $\Pi \in \Lambda(\overline{\Pi})$ is a sequential plan for the task. Intuitively, a partial-order plan represents a family of related sequential plans.

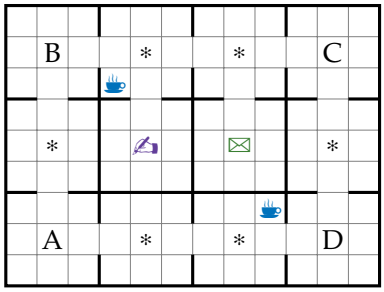


Figure 1: The OFFICEWORLD.

Running Example We consider a version of the OFFICEWORLD domain described by Toro Icarte et al. [10]. The low-level environment is represented by the grid displayed in Figure 1. An agent situated in any cell can try to move in any of the four cardinal directions, succeeding only if the movement does not go through a wall. The symbols in the grid represent events that can be perceived by the agent: it picks up coffee or mail when it reaches the locations marked with blue cups or the green envelope, respectively, or it can deliver what it picked up by reaching the office, marked by the purple writing hand, etc. The locations marked ✻ are places the agent must not step on. The four additional named locations (A, B, C, D) can also be recognized by the agent. An example of a task is that of delivering both mail and coffee to the office. For this task, any optimal policy will need to choose whether to get the coffee or mail first depending on the agent's initial position.

4 Planning Models in RL

Given a low-level environment $\mathcal{E} = \langle S, A, p \rangle$ and the set of associated symbols $\mathcal{P}$ that represent possible observations, a symbolic model for $\mathcal{E}$ is specified as $\mathcal{M} = \langle \mathcal{D}, \alpha \rangle$, where $\mathcal{D} = \langle F, \mathcal{A} \rangle$ is a planning domain and $\alpha\colon \mathcal{A} \to 2^{\mathcal{P}}$ is a function that associates planning actions with sets of observations.

Note that the symbols in F do not directly map to the observations, and that they can therefore be used to model state properties that cannot be directly perceived in the low-level environment. In the OFFICEWORLD running example, some of the symbols in F represent state properties that are non-Markovian in the low-level environment, such as the set of locations that were visited in the past, and whether or not the agent is carrying coffee or mail.

Note that one of the key properties of automated planning systems based on symbolic models is that specifying individual tasks is a very simple problem. We take advantage of this by enabling the description of such tasks—goals in the symbolic models—to be communicated to an RL agent.

From Symbolic Planning Actions to Options Given an environment $\mathcal{E}$ and a corresponding symbolic model $\mathcal{M}$, we can define a set of options that represents relevant transitions that can occur in $\mathcal{M}$. For every planning action $a \in \mathcal{A}$, we build a target set T_a comprised of all the low-level states in which every observation associated with a is perceived. Formally, this is defined in terms of L^{-1}, the preimage of the labeling function: $T_a = L^{-1}(\alpha(a)) \subseteq S$. Then, the set of options can be defined as $O(\mathcal{M}) = \{\langle \pi_a, T_a \rangle \mid a \in \mathcal{A}\}$, where every π_a is a policy trained specifically for reaching the states in T_a. Note that two or more actions in $\mathcal{A}$ may produce the same target set and be represented by a single option.

From Plans to Meta-Controllers We propose that a good way of communicating the high-level tasks to the agent is not in terms of the goals of the task, but rather in terms of high-level plans that achieve them. This allows us to specify tasks that are non-Markovian from the perspective of the low-level environment, such as delivering coffee in the OFFICEWORLD, while simultaneously enabling a natural form of task decomposition.

To actually give a specific high-level plan to an RL agent, we will design a simple reward machine that directly represents the execution of the plan. Reward of 1 will be given only upon completing this execution.

For a sequential plan $\Pi = [a_0, a_1, \ldots, a_n]$, the implementation of the corresponding reward machine is very straightforward and shown in Figure 2a. If instead of a sequential plan we consider a partial-order plan $\overline{\Pi} = \langle \overline{\mathcal{A}}, \prec \rangle$, we need a reward machine that effectively represents all possible orderings of the plan, in a way that any execution of one of them results in receiving a reward of 1 only on the last step. We build this machine by including one state for every possible subset of $\overline{\mathcal{A}}$. This ensures that the machine's executions correctly keep track of which actions have already taken place. Note that some of these states will not be reachable. In practice, we will only generate the states as needed. The possible transitions will be defined by explicitly considering $\Lambda(\overline{\Pi})$, as described in Figure 2b.

- $U = \{u_0, u_1, \ldots, u_{n+1}\}$
- $\delta_r(u_i, u_j) =$
  $\begin{cases} 1 & \text{if } i = n \text{ and } j = n+1 \\ 0 & \text{in any other case} \end{cases}$
- $\delta_u(u_i, P) =$
  $\begin{cases} u_{i+1} & \text{if } P = \alpha(a_i) \\ u_i & \text{in any other case} \end{cases}$

(a) Reward machine based on a sequential plan $\Pi = [a_0, a_1, \ldots, a_n]$.

- $U = 2^{\overline{\mathcal{A}}}$, $u_0 = \varnothing$
- $\delta_r(u_i, u_j) = \begin{cases} 1 & \text{if } u_j = \overline{\mathcal{A}} \\ 0 & \text{in any other case} \end{cases}$
- $\delta_u(u, P) =$
  $\begin{cases} u \cup \{a\} & \text{if } \exists a \in \overline{\mathcal{A}}, \Pi \in \Lambda(\overline{\Pi}), \\ & \quad \text{s.t. } u = \textbf{prefix}(\Pi, a) \\ & \quad \text{and } P = \alpha(a) \\ u & \text{in any other case} \end{cases}$

(b) Reward machine based on a partial-order plan $\overline{\Pi} = \langle \overline{\mathcal{A}}, \prec \rangle$.

Figure 2: Reward machines defined by plans.

Note that the given definition of δ_u does not necessarily imply a function. Consider two actions $a, b \in \overline{\mathcal{A}}$ such that $P = \alpha(a) = \alpha(b)$, that is, actions that result in the same observations. In the OFFICEWORLD, an action to simply visit the office achieves the same event as the action to deliver coffee. Now, if there are two linearizations of $\overline{\Pi}$, Π_a and Π_b, such that $u = \textbf{prefix}(\Pi_a, a) = \textbf{prefix}(\Pi_b, b)$, then either action could be used in the definition of $\delta_u(u, P)$. In such cases, we arbitrarily choose which of the actions to use, as either choice represents a valid linearization of $\overline{\Pi}$.

From Reward Machines to Meta-Controllers Given an environment $\mathcal{E}$ and a symbolic model $\mathcal{M} = \langle \mathcal{D}, \alpha \rangle$, we want to leverage the set of options $O(\mathcal{M})$ to efficiently find a low-level policy for achieving a high-level goal g defined in terms of $\mathcal{D}$. A naive approach consists of finding a sequential plan for g and then attempting to execute the corresponding options in order. A more flexible approach is to learn a meta-controller over the options while using the plan only to specify the reward function. The latter approach also allows the use of a partial-order plan, resulting in a less restrictive reward.

Our third and main approach combines the flexibility of partial-order plans with the simplicity of naively following a plan. Effectively, we train a meta-controller that is a priori restricted to only select among options that correspond to actions that are reasonable within the context of a partial-order plan. In practice, this means that the meta-controller is limited to selecting options that can advance the reward machine toward its goal, an idea introduced by Toro Icarte et al. as a simple way of exploiting the structure in a reward machine [10] .

5 Empirical Evaluation

We evaluated our approach by considering two different low-level environments and respective high-level symbolic models. In each case, we defined a set of option candidates using the approach outlined in Section 4. We subsequently pretrained these options, and finally evaluated our approach on a number of new tasks for each environment. For each task we computed sequential and partial-order plans, and built the corresponding reward machines. We compare our results against the use of the same pretrained options in a standard HRL framework, and against the naive approach outlined above. In all cases, the training was done through the Q-learning algorithm.

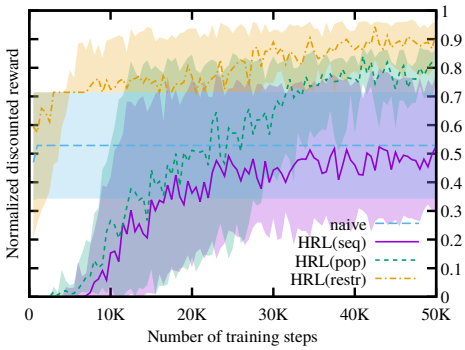
The first test environment is the OFFICEWORLD running example. Here the high-level actions include visiting any of the named locations, getting coffee, getting mail, delivering either coffee or mail to the office, and delivering both coffee *and* mail to the office. This results in options for going to any of the named locations or to the office, and for getting coffee and getting mail. The actions for delivering to the office are all mapped to the same observation, which occurs whenever the agent reaches the office, and therefore correspond to the same option. For this environment, we tested on tasks consisting of 6 different goals and 10 random initial states for each goal.

Our second environment is the Minecraft-inspired gridworld described by Andreas et al. [1]. The grid contains raw materials (e.g., wood, iron) and locations where the agent can combine materials to produce refined materials (e.g., wooden planks), tools (e.g., hammer), and goods (e.g., goldware). The high-level actions allow for collecting each of the raw materials, and for achieving the combinations. The types of tasks that we evaluated on include examples such as *"build a bridge"* or *"have a hammer and a pickaxe"*. We ran experiments on 10 different maps, 10 different final-state goals, and 4 random initial states for each combination of map and goal.

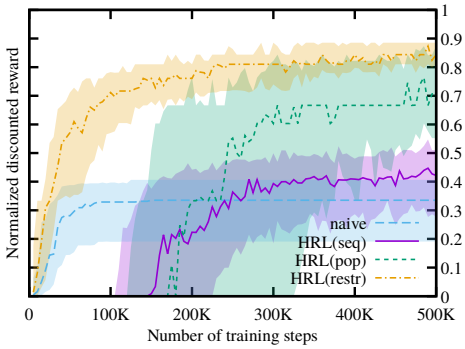
In both environments, we pretrained options by generating a small set of random initial states. We simultaneously trained a single policy for each option. The experience observed when training for any given option was used for training all other options, in an off-policy learning setting. This pretraining was restricted to a small number of reinforcement learning steps. In each independent experiment, option policies were continuously refined as the agent continued to interact with the environment.

We used the Fast Downward planning system [4] to quickly produce high-quality sequential plans. To get partial-order plans, we relaxed the sequential plans by solving a mixed integer linear program, following Muise et al. [8].

Results and Discussion We report the results for the OFFICEWORLD and MINECRAFTWORLD environments in Figures 3a and 3b. Each graph displays the performance obtained after training with the labeled algorithm for the specified number of steps. HRL(restr) refers to our main approach; HRL(seq) and HRL(pop) refer to the standard HRL approach, respectively using the rewards specified by the sequential and partial-order plans. In all cases, we report the normalized median discounted reward (using discount $\gamma = 0.9$) obtained on all tasks and several independent trials. We also show the median quintile performance (shaded area). Rewards were normalized by the best value obtained for each task across all independent experiments.



(a) OFFICEWORLD



(b) MINECRAFTWORLD

Figure 3: Experimental performance obtained on previously unseen tasks.

The aim of our experiments was to validate our approach as a method for easily solving unseen tasks. In particular, they show how by just having pretrained options and a high-level plan, we could obtain high-quality policies for the new tasks with little extra learning. Our results were positive and show that the approaches that restrict which options can be used based on the actions presented by the plan produced good policies after very few training steps. The naive method quickly reached a plateau in its performance; but even when compared to this, the other methods we tested needed up to an order of magnitude more training steps to reach comparable results. Our main approach significantly outperformed all other approaches, even after a large number of training steps.

To conclude, we believe that the automatic generation of goal-relevant options and ordering constraints over them—in conjunction with the ability to explicitly represent them in a structured reward function—is one of the key aspects that will enable RL systems to be both taskable and scalable. Moreover, we believe that symbolic planning and knowledge representation techniques can provide the ideal framework for generating such options and constraints. Planning formalisms and algorithms support a plethora of different domain and solution properties, such as temporally-extended goals, preferences, diverse plans, numeric constraints, and more. All of these could be used for representing rich reinforcement learning tasks, while preserving useful structure that can be exploited when learning policies.

References

[1] J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. In *ICML*, volume 70 of *PMLR*, pages 166–175, 2017.

[2] M. J. Grounds and D. Kudenko. Combining reinforcement learning with symbolic planning. In *AAMAS III*, volume 4865 of *LNCS*, pages 75–86, 2008.

[3] M. Grześ and D. Kudenko. Plan-based reward shaping for reinforcement learning. In *IS*, volume 2, pages 10-22–10-29, 2008.

[4] M. Helmert. The Fast Downward planning system. *JAIR*, 26:191–246, 2006.

[5] S. James, B. Rosman, and G. Konidaris. Learning to plan with portable symbols. In *PAL@ICML/IJCAI/AAMAS*, 2018.

[6] G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *JAIR*, 61:215–289, 2018.

[7] D. Lyu, F. Yang, B. Liu, and S. Gustafson. SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *AAAI*, 2019.

[8] C. J. Muise, J. C. Beck, and S. A. McIlraith. Optimal partial-order plan relaxation via MaxSAT. *JAIR*, 57:113–149, 2016.

[9] R. S. Sutton, D. Precup, and S. P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *AIJ*, 112(1-2):181–211, 1999.

[10] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *ICML*, volume 80 of *PMLR*, pages 2112–2121, 2018.

[11] F. Yang, D. Lyu, B. Liu, and S. Gustafson. PEORL: integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In *IJCAI*, pages 4860–4866, 2018.

[12] A. Zhang, S. Sukhbaatar, A. Lerer, A. Szlam, and R. Fergus. Composable planning with attributes. In *ICML*, volume 80 of *PMLR*, pages 5837–5846, 2018.

Doubly Robust Estimators in Off-Policy Actor-Critic Algorithms

Riashat Islam[1*] **Samin Yeasar Arnob**[2*] **Doina Precup**[1]

[1]Mila - McGill University
[2]McGill University

Abstract

Off-policy learning in deep reinforcement learning (RL) relies on the ability of using past samples from an experience replay buffer, where samples are collected by a different behaviour policy. Despite its usefulness in terms of sample efficiency, off-policy learning often suffer from high variance. Doubly robust (DR) estimators were introduced by Jiang and Li [2016], Thomas and Brunskill [2016] to guarantee unbiased and lower variance estimates in off-policy evaluation. In this work, we extend the idea of doubly robust estimation in actor-critic algorithms, to achieve low variance estimates in the off-policy critic evaluation. DR estimators can also be interpreted as a control variate Thomas and Brunskill [2016]. In policy gradient algorithms relying on gradient of action-value function, we therefore propose using a control variate to achieve lower variance in the critic estimate itself. We suggest that controlling the variance, and achieving a good estimate of the action-value function is a key step towards stability of policy gradient algorithms. We demonstrate the usefulness of using a doubly robust estimator in the policy critic evaluation in a popular off-policy actor-critic algorithm on several continuous control tasks. Extending the usefulness of DR estimators in policy gradients may be an important step towards improving stability of policy gradient methods, relying on reliably good, unbiased and low variance estimations of the critic.

Keywords: Off-policy learning, Policy gradient and Actor-Critic algorithms, Doubly Robust Estimators, Control Variates

[*]Equal contribution. Correspondence: {riashat.islam}@mail.mcgill.ca.

1 Introduction

Policy gradient methods are a wide class of model-free algorithms used to solve continuous control tasks Lillicrap et al. [2015], Silver et al. [2014a]. These class of algorithms are adaptable to solve a range of complex tasks. However, due to the instability of policy gradient algorithms, especially in the off-policy case, as noted by Henderson et al. [2018], often state dependent or state-action dependent baselines are used to reduce variance of the policy gradient estimators. Monte-Carlo on-policy policy gradient estimates Williams [1992] often suffer from high variance, and requires state dependent unbiased baselines to reduce variance. Actor-Critic algorithms replaces the Monte-Carlo returns with an estimated return to reduce variance, but at the cost of introducing bias. A wide range of state-action dependent baselines have also been introduced, which are often mostly used for off-policy actor-critic algorithms Lillicrap et al. [2015]. Although several state-action dependent baselines have been introduced Gu et al. [2016], there exists a mirage between the choice of the baseline Tucker et al. [2018], and the choice of the right baseline is often not clear. It is often argued that the variance of action dependent baselines often increases, compared to using a state dependent baselines, in few of the standard control tasks Tucker et al. [2018].

In this paper, we extend the idea of doubly robust off-policy evaluation Jiang and Li [2016] in actor-critic algorithms. Doubly robust (DR) estimators have better theoretical understandings, both from the bandits literature Dudík et al. [2015] and for sequential decision making Jiang and Li [2016]. Thomas and Brunskill [2016] also extends the idea of doubly robust off-policy evaluation as a control variate to further reduce variance while keeping the estimators unbiased for both finite and infinite horizon settings. We propose to extend doubly robust estimators in the off-policy critic evaluation of actor-critic algorithms. More interestingly, the introduction of DR estimators can be thought of as an using an advantage function, $\hat{A}$ in the critic estimate itself, as we derive later, which has an interesting connection to control variates for critic estimation to reduce variance of regression estimate. Instead of using a control variate in the gradient estimator, we are subtracting a state dependent baseline, and adding a state-action dependent control variate in the critic estimate itself. Through the use of a separate reward function approximator, we suggest that a DR estimator which adds a state-action dependent baseline while subtracting a state dependent baseline can make off-policy gradient algorithms work significantly better in practice. Importantly, we believe that extending the idea of DR estimators to actor-critic algorithms is an interesting step in itself.

2 Preliminaries

In policy gradient methods, the aim is to learn a parameterized policy $\pi_\theta(a|s)$ to maximize the discounted sum of cumulative returns along the sampled trajectories, given by $J(\pi_\theta) = \mathbb{E}_{\pi_\theta}[\sum_{i=t+1}^{\infty} \gamma^i r(s_i, a_i)]$. Based on the policy gradient theorem Sutton et al. [2000], we can improve the policy parameters θ using the policy gradient, which can be computed with Monte-Carlo estimation $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)]$, where, the $\mathbb{E}$ above uses samples under the current policy π. Often the policy gradient estimator can suffer from high variance, and hence an advantage function $A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$ or a state dependent baseline, or a combination of both is used in practice to reduce baseline $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)(Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s))]$. Alternatively, often the gradient is also estimated with an advantage function critic and using a state-action dependent baseline, such that $A^w(s, a) - Q(s, a)$ where the critic uses a function approximator parameterized by w and a separately learned baseline is used.

2.1 Off-Policy Actor-Critic Algorithms

Instead of on-policy gradient estimators, which can be sample-inefficient in practice, due to their inability to re-use data from past experiences, often an off-policy gradient estimate is preferred, based on the deterministic policy gradient (DPG) theorem Silver et al. [2014a]. For continuous control tasks, the DDPG algorithm Lillicrap et al. [2015] is often used due to their ease ability to learn from experience replay buffer. The off-policy policy gradient estimator is given by $\nabla_\theta J(\theta) = \mathbb{E}_\mu[\nabla_\theta Q^w(s, \pi_\theta(s)]$, where Q^w is a critic estimate and π_θ is a deterministic policy which outputs continuous actions, to allow directly finding the gradient of the action-value function. Due to the instability of off-policy gradient methods with function approximators, we often require careful fine-tuning of this algorithm Henderson et al. [2018] as the gradient estimate is directly related to the estimate of the critic. Since the DPG Silver et al. [2014a] algorithm can avoid importance sampling (IS) corrections, typically required in off-policy learning Precup [2000], the critic can be evaluated with a regression based objective without any high variance IS corrections being required $L(w) = \mathbb{E}_\mu[(r(s, a) + \gamma Q(s', \pi_\theta(s')) - Q(s, \pi_\theta(s)))^2]$, where the $\mathbb{E}_\mu$ is under samples from the experience replay buffer, and the off-policy critic evaluation is a regular one-step temporal difference (TD) based update without requiring off-policy corrections, even though we are using old samples from the replay buffer. This is due to the DPG theorem Silver et al. [2014b] which avoids an integral over the action space, avoiding the need for IS corrections. However, the DDPG algorithm can often be unstable to use in practice Henderson et al. [2018], and a state-action dependent or state dependent baseline can be used in the gradient estimate.

In this work, we instead argue that a control variate in the form of a doubly robust estimator Thomas and Brunskill [2016] should instead be used in the off-policy critic evaluation of DDPG. Since the gradient estimate is dependent on

directly finding gradient of action-value function, it is better to use a baseline which is unbiased but reduces variance in the off-policy critic evaluation itself.

2.2 Doubly Robust Off-Policy Evaluation

The doubly robust (DR) estimator in MDPs was introduced by Jiang and Li [2016], Thomas and Brunskill [2016] to reduce the variance of off-policy evaluation, while keeping the regression based estimators unbiased. Jiang and Li [2016] argued that instead of using importance sampling corrections, which is unbiased, but can have high variance, it is better to use DR estimators in off-policy evaluation tasks. The key step in DR estimator is to use the following unbiased estimator

$$V_{DR}(s) = \hat{V}(s) + \rho[r(s,a) + \gamma V_{DR}(s') - \hat{Q}(s,a)] \tag{1}$$

where in the case of off-policy, ρ is the importance sampling correction in the update, and we replace V^π with V^π_{DR} to denote a DR estimation of the off-policy evaluation. A key requirement in DR estimators is to use an approximation to the MDP model since the $\hat{V}$ requires the rewards from an approximation of the MDP. In other words, $\hat{R}$ used to compute $\hat{V}$ is the model's prediction of the reward. Given the samples from past data and an approximate model of the MDP, the goal of DR estimators is to produce a low variance regression mean equated error estimate $MSE(V_{DR}, V^\pi)$.

3 Approach

In this work, we extend the idea of DR estimators in the off-policy DDPG algorithm, where we propose to use a doubly robust estimator, as a control variate, to reduce variance of the critic estimates. The key idea of our work is to use a low variance and unbiased DR estimator to estimate the critic, instead of using a control variate in the gradient estimate itself. Our key intuition is as follows : since the policy gradient estimate in DDPG relies directly on finding gradient of action-value function, the important quantity in DDPG is to have a reliably good low variance estimate of the critic. We therefore aim to reduce the variance of the critic estimate itself. Since we can re-use samples from the experience buffer, we can use an off-policy low variance evaluation of the critic based on sampled states and actions to further reduce the variance of the policy gradient estimator in the actor update. When we incorporate the DR estimator, the off-policy critic estimate in DDPG $Q_{DR}(s,a)$ will be as follows

$$Q_{DR}(s,a) = \hat{Q}(s,a) + \Big[r(s,a) + \gamma Q_{DR}(s', \pi_\theta(s')) - \hat{V}(s)\Big] \tag{2}$$

where $\hat{Q}$ and $\hat{V}$ are computed based on an approximation of the rewards $\hat{R}$ using a *reward function approximator*.

Reward Function Approximator : We extend the idea of DR estimator Jiang and Li [2016] in actor-critic algorithms in deep reinforcement learning. Typically DR estimation requires an approximate MDP model to have an estimate of the rewards $\hat{R}$. In this work, instead of using an approximate model, we instead use a parameterized reward function approximator $\hat{R} = f_\phi(s,a)$ to predict the rewards. In other words, we use an approximation $\hat{R}$ of the true reward function $r(s,a)$ and learn the reward function approximator based on sampled states from the experience replay buffer. This is a supervised learning (SL) step, where given the s, a, r, s' tuples in the reply buffer, we can train an approximate reward model $\hat{R}$ to predict the true rewards r based on the following regression based update

$$L(\phi) = \mathbb{E}_{s,a,r\sim Buffer}[(\hat{R}(s,a) - r(s,a))^2] \tag{3}$$

The reward function approximator $f_\phi(s,a)$ therefore predicts the reward for the batch of experiences available in the replay buffer. The predicted rewards $\hat{R}$ can then be used to compute $\hat{Q}$, which is a separate control variate or baseline function approximator. The $\hat{Q}$ can further be learned, as typically done for a separate control variate as follows

$$L(\hat{Q}) = \mathbb{E}_{s,a,r,s'\sim Buffer}[(\hat{R}(s,a) + \gamma\hat{Q}(s', \pi_\theta(s')) - \hat{Q}(s,a))^2] \tag{4}$$

Following 4 we can further estimate $\hat{V}(s)$ similar update rule. Therefore, using a reward function approximator, we can compute Q_{DR} as in doubly robust regression estimation, without explicitly using an approximation of the model of the MDP.

For learning the off-policy critic estimate in DDPG, we therefore use the following update rule, based on the DR estimator, where the Q_{DR} is estimated with a parameterized function approximator w.

$$L(w) = \mathbb{E}_{s,a,r,s'\sim Buffer}\Big[\hat{Q}(s,a) + \big(r(s,a) + \gamma Q_{DR}(s', \pi_\theta(s')\big) - \hat{V}(s) - Q_{DR}(s,a)\Big] \tag{5}$$

Equation 5 can be explained as follows. Instead of the regular critic update in DDPG, we have introduced the DR estimate of the critic, where $\hat{Q}$ and $\hat{V}$ are estimated based on a reward function approximator. Equation 5 can be interpretated as introducing a DR control variate Thomas and Brunskill [2016] in the critic update, where we have subtracted a state dependent baseline (as typically done to reduce variance of policy gradient estimates) and added a state-action dependent control variate (unlike typically subtractinga this control variate). As in DR off-policy evaluation, this also achieves an unbiased, low variance estimate of the critic in actor-critic algorithms. This can also be interpreted as adding a separate advantage function estimation since $\hat{A} = \hat{Q} - \hat{V}$, where the advantage function is now learnt with a separate approximation of the rewards $\hat{R}$. In our experiments, we show that adding a state-action dependent inverse baseline (since we are adding the control variate), inspired from using DR estimation for off-policy evaluation, we can significantly improve our estimate the critic (lower MSE) and improve performance of our algorithm for continuous control tasks.

4 Experimental Results

We evaluate the performance of our DDPG algorithm with a doubly robust critic estimator on several continuous control Mujoco tasks Todorov et al. [2012]. We use two variants of the DR estimator in the critic (a) : We add only a state-action dependent $\hat{Q}(s,a)$ in the critic estimate, denoted by DDPG-HDR and (b) use the full DR estimator as the advantage term $\hat{Q} - \hat{V}$). Experiments are evaluated on the Half-Cheetah-v2, Walker-2d-v2, Hopper-v2, Swimmer-v2, InvertedPendulum-v2, InvertedDoublePendulum-v2 environments, and evaluated an average over 3 runs with random seeds.

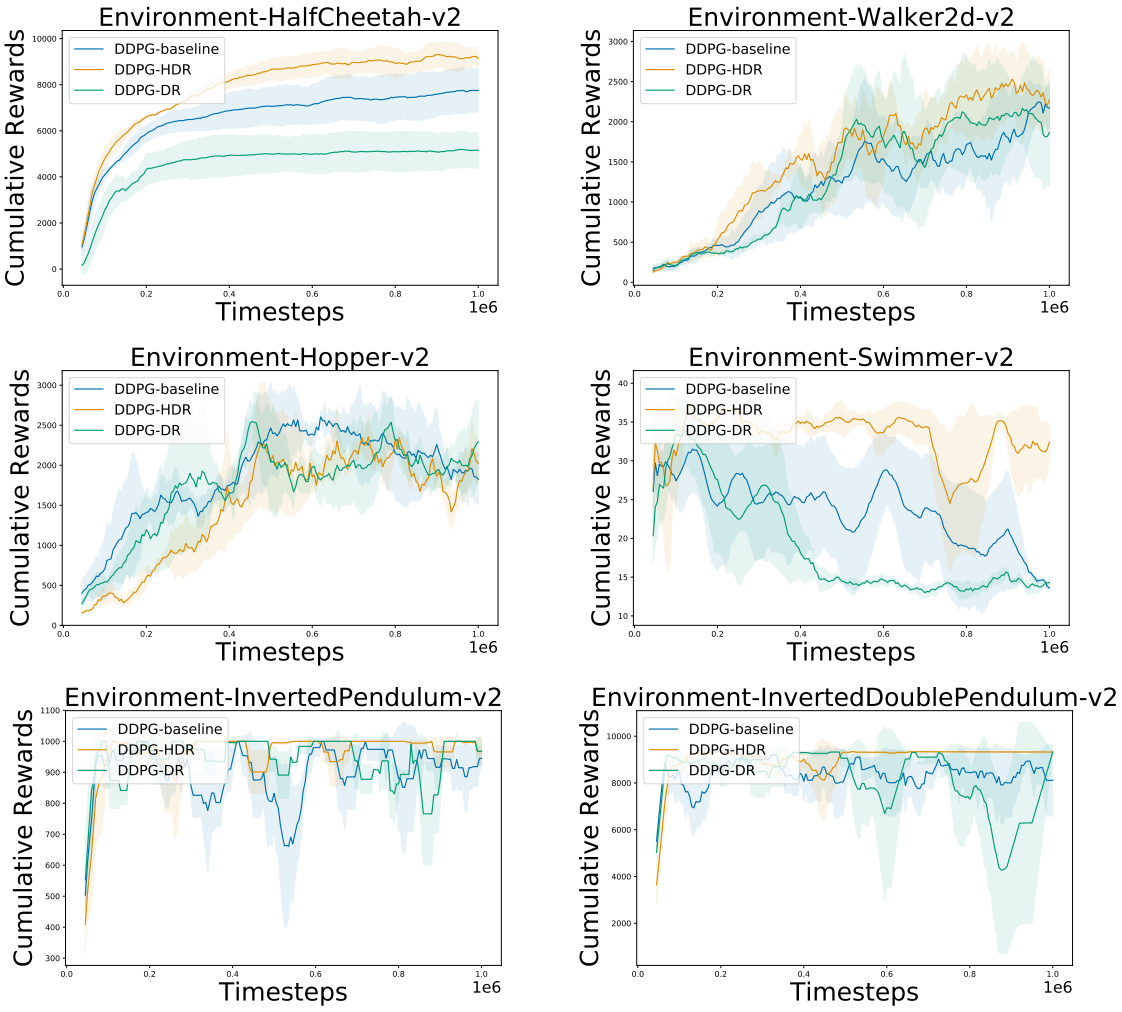


Figure 1: Performance Comparison of DDPG and Variants of Doubly Robust DDPG

Figure 1 shows that using a DR estimator in DDPG, we can significantly reduce variance and improve the performance of our algorithm over the standard baseline over all the environments except Hopper-v2. Interestingly, however, using the

full form of the DR estimator does not necessarily lead to better performance, but adding only a state-action dependent control variate in the critic estimate has a better effect, as shown in figure 1.

5 Discussion and Conclusion

In this work, we have shown that the doubly robust (DR) estimator can be extended to the actor-critic setting, where the critic can be estimated with a DR estimator to reduce variance while keeping the estimator unbiased. Even though the DR estimator depends on an approximate model of the MDP, we suggest that to extend DR to be fully model-free, we can instead use a reward function approximator f_ϕ and learn it to predict the rewards $\hat{R}$ as an approximation to the estimated model rewards. The reward function approximator can be learnt using the same batch of data in the experience replay buffer without more computation. We evaluated the performance of our algorithm on standard baselines, building off the DDPG algorithm, and showed that the introduction of a control variate in the critic estimate can lead to marginally better performance.

In future work, it would be interesting to see other ways the reward function approximator can be used to predict the rewards $\hat{R}$. Since predicting the rewards can be considered as a supervised learning problem, it would be interesting to see the effect of over-fitting in the reward function approximation. Furthermore, we would require theoretical analysis of the bias variance trade-off of the DR estimator in the actor update, to fully understand the potential of DR estimators in the actor-critic setting. We evaluated our variance reduction on DDPG only. More experiment are required to show, how well does the variance reduction of DR estimators generalize in policy gradient methods.

References

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 652–661, 2016. URL http://jmlr.org/proceedings/papers/v48/jiang16.html.

Philip S. Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2139–2148, 2016. URL http://jmlr.org/proceedings/papers/v48/thomasa16.html.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014a.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3207–3214, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16669.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

Shixiang Gu, Timothy P. Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *CoRR*, abs/1611.02247, 2016. URL http://arxiv.org/abs/1611.02247.

George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E. Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 5022–5031, 2018. URL http://proceedings.mlr.press/v80/tucker18a.html.

Miroslav Dudík, Dumitru Erhan, John Langford, and Lihong Li. Doubly robust policy evaluation and optimization. *CoRR*, abs/1503.02834, 2015. URL http://arxiv.org/abs/1503.02834.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 2000.

Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014b.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Off-Policy Policy Gradient Theorem with Logarithmic Mappings

Riashat Islam
Mila - McGill University
riashat.islam@mail.mcgill.ca

Zafarali Ahmed
Mila - McGill University
zafarali.ahmed@mail.mcgill.ca

Pierre-Luc Bacon
Stanford University
plbacon@cs.stanford.edu

Doina Precup
Mila - McGill University
dprecup@cs.mcgill.ca

Abstract

Policy gradient theorem [1] is a fundamental result in reinforcement learning. A class of continuous control tasks rely on policy gradient methods. However, most of these algorithms rely on using samples collected on-policy. While there are serveral approaches proposed for off-policy policy gradients, there exists a lack of an *off-policy gradient theorem* which can be adapted for deep reinforcement learning tasks. Often off-policy gradient methods are difficult to use in practice due to the need for an importance sampling correction which can have unbounded variance. In this paper, we propose a derivation for an off-policy policy gradient theorem which can completely avoid high variance importance sampling corrections. Towards this goal, we introduce the existence of a policy gradient theorem using a non-linear Bellman equation (logarithmic mappings of value function). We show that using logarithmic mappings of the policy gradient objective, we achieve a lower bound to the policy gradient, but can avoid importance sampling and derive the gradient estimate where experiences are sampled under a behaviour policy. We further develop an off-policy actor-critic algorithm, and suggest that the proposed off-policy gradient can be used for deep reinforcement learning tasks, for both discrete and continuous action spaces.

Keywords: Reinforcement learning; Policy Gradient Methods; Off-Policy Learning; Actor-Critic Algorithms

Acknowledgements

1 Introduction

Off-policy learning in reinforcement learning (RL) is often desired due to its ability to learn from and re-use multiple streams of experiences collected under a behaviour policy, to learn about a target policy. Off-policy RL is often desired due to their sample efficiency, and scaling up off-policy methods has been a fundamental goal towards efficient RL. A central approach to off-policy learning, however, relies on importance sampling (IS) corrections [2], which can exhibit high variance, making off-policy algorithms often unstable to use in practice.

A wide range of on-policy [3, 4] and off-policy [5, 6] algorithms exist. Unfortunately, the choice of which algorithm to use is often dictated by the action space being discrete or continuous. In particular, the range of off-policy gradient algorithms for discrete action spaces is quite limited. Previously, [7] proposed the off-policy counterpart of the policy gradient theorem [1], but uses a crude approximation to the gradient term and the theorem only works for tabular representations of the policy. [8] proposed an actor-critic algorithm under off-policy training, and shows that there exists convergence guarantees for the case where the critic uses a linear function approximation. Recently, [9] proposed the off-policy gradient theorem, for the case of both stochastic and deterministic policies, but relies on estimating the emphatic weightings which makes the algorithm often difficult to scale in practice. The success of off-policy policy gradients is widely dominated due to the existence of the Deterministic Policy Gradient theorem [10], which only works for the case of deterministic policies and continuous actions.

In this work, we propose an *off-policy policy gradient theorem* following the policy gradient theorem [1] and suggest the provide the following contributions. We derive the policy gradient estimates with logarithmic mappings assuming positive rewards, but the results can be extended for general rewards by using a decomposition of the value function under positive and negative rewards.

- We derive a policy gradient theorem based on non-linear logarithmic mappings of the value function. We first derive the policy gradient, and then extend the result for the off-policy case which has a desriable property as follows
- The off-policy policy gradient theorem using logarithmic mappings can avoid importance sampling (IS) corrections which can have high unbounded variance. This is an important result in off-policy RL, since most of off-policy learning methods relies on importance sampling and variants
- We suggest that by using Jensen's inequality with logarithmic mappings of the policy gradient objective, we can compute the Monte-Carlo estimate of the gradient under samples from the behaviour policy, without the need for any importance sampling corrections
- The proposed off-policy gradient objective can be particularly interesting as it can be extended for both discrete and continuous action spaces - a result lacking in policy gradients literature, since the choice of the algorithm often depends on the action space
- We derive an off-policy actor-critic algorithm, where the critic can be estimated with logarithmic mappings off-policy and can avoid direct dependency on importance sampling corrections in its estimate. We can show existence of a unique fixed point for the off-policy one-step critic update with logarithmic mappings.

2 Preliminaries and Background

The goal of reinforcement learning is to find a policy π that maximizes the expected return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$. In policy gradient methods, we consider parameterized policies $\pi_\theta(a|s)$, and optimize the objective by adjusting the policy parameters θ based on the policy gradient theorem [1, 11]: $\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\pi(s),a\sim\pi(a|s)}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)]$. Note here that the expectation is under the current policy π, and requires trajectories to be sampled under the current policy to obtain a sample based estimate. Several variants of the policy gradient theorem have been proposed, including Trust Region Policy Optimization (TRPO) which often dominates the success of these algorithms in continuous control tasks. However, most of these algorithms lack the ability to re-use past experiences or off-policy data. In off-policy learning, importance sampling corrections are used [2] to correct for the difference in distributions and learn the value functions from experiences that are gathered under a behaviour policy $\mu(a|s)$. However, since policy gradient methods are based on returns until end of the trajectory, this would require per step importance sampling corrections, and therefore the product of IS corrections would exhibit high variance, making it unsuitable for solving complex tasks. Off-PAC [7] proposed using marginal importance weights to reduce the variance, but still requires estimating the IS ratio, and further approximates the gradient by dropping a term. Recent work from [9] tries to address this issue by taking the full approximation to the gradient, but instead requires estimating the emphatic weightings For deep reinforcement learning, perhaps the most successful off-policy policy gradient algorithm is the Deep Determininistic Policy Gradient (DDPG), derived from the deterministic policy gradient theorem, but is only known to work for deterministic continuous action policies. Although variants of the DDPG algorithm have been proposed, as in the Soft-Actor-Critic (SAC), and the ACER algorithm, but all these derives from the either the Off-PAC or the DPG policy gradient theorem.

3 Policy Gradient Theorem with Logarithmic Value Functions

In this section, we show that by considering logarithmic mappings of the value function $\log V^\pi(s)$, we can derive the existence of a policy gradient theorem. We first show that a policy gradient theorem exists for logarithmic value functions, and then derive the significance of using $\log V^\pi(s)$ to derive the off-policy counterpart of the policy gradient theorem. This has a particularly interesting form as for the off-policy case, unlike existing off-policy policy gradient theorem as in [7, 9], we can completely avoid the dependency on the importance sampling corrections, which is can have unbounded variance. This is inspired by the idea of having a non-linear Bellman equation for value functions with logarithmic mappings where we can write the value function as follows.

Let us first consider the case where the reward function, and hence the value functions are positive. Consider the logarithmic mapping $f(x) = \ln(x)$ and its inverse, $f^{-1}(x) = \exp(x)$. We can define an update for Q directly in the logarithmic space by applying the mapping $\log Q(s,a) = \tilde{Q}(s,a)$. We can therefore write the update equation as $\tilde{Q}(s,a) := (1-\alpha)\tilde{Q}(s,a) + \alpha f[r(s,a) + \gamma f^{-1}(\tilde{Q}(s,a))]$, where $\tilde{Q}(s,a)$ is an estimate of the expected return mapped to the logarithmic space and we can retrieve the regular $Q(s,a)$ with the inverse mapping $f^{-1}(\tilde{Q}(s,a))$ [1]. Recent work also demonstrates the existence of logarithmic value functions (under submission).

Following [1], we derive the policy gradient theorem for logarithmic mappings of the value function. We prove the theorem for the start-state formulation. In the next section, we will show that a similar form of the policy gradient with logarithmic mappings would hold even for the case of off-policy learning, with the only difference being the policy under which samples are collected.

The policy gradient objective is given by $J(\pi) = \mathbb{E}_{s\sim d_\pi(s)}[V^\pi(s)]$. Taking $\log$ on both sides, and applying Jensen's inequality, we get a lower bound to the policy gradient objective $\tilde{J}(\pi)$, but also get a logarithmic mapping of $V^\pi(s)$, ie, $\log J(\pi) \geq \mathbb{E}_{s\sim d_\pi(s)}[\log V^\pi(s)]$. Assumiing policy parameterization $\pi_\theta(a|s)$, we want to find $\nabla_\theta \log V^{\pi_\theta}(s)$.

$$\nabla_\theta \log V^\pi(s) = \nabla_\theta \log[\sum_a \pi(a,s) Q^\pi(s,a)] = \nabla_\theta[\log[\mathbb{E}_{a\sim\pi(a|s)}\, Q^\pi(s,a)]] \tag{1}$$

$$>= \nabla_\theta[\mathbb{E}_{a\sim\pi(a|s)}[\log Q^\pi(s,a)]] = \nabla_\theta[\sum_a \pi(a|s) \log Q^\pi(s,a)] \tag{2}$$

$$= \sum_a [\nabla_\theta \pi(a|s) \log Q^\pi(s,a) + \pi(a|s) \nabla_\theta \log Q^\pi(s,a)] \tag{3}$$

$$= \mathbb{E}_{a\sim\pi(a|s)}[\nabla_\theta \log \pi(a|s) \log Q^\pi(s,a) + \nabla_\theta \log Q^\pi(s,a)] \tag{4}$$

Taking $\mathbb{E}_{s\sim d_\pi(s)}$ on both sides, we therefore get the *policy gradient theorem with logarithmic value functions*, as in equation 5.

$$\sum_s d_\pi(s)[\nabla_\theta \log V^\pi(s)] = \sum_s d_\pi(s)[\mathbb{E}_{a\sim\pi(a|s)}[\nabla_\theta \log \pi(a|s) \log Q^\pi(s,a) + \nabla_\theta \log Q^\pi(s,a)]] \tag{5}$$

$$= \mathbb{E}_{s\sim d_\pi(s), a\sim\pi(a|s)}[\nabla_\theta \log \pi(a|s) \log Q^\pi(s,a) + \nabla_\theta \log Q^\pi(s,a)]] \tag{6}$$

We can interpret the above policy gradient theorem for logarithmic mappings as follows. This result is similar to the policy gradient theorem [1], except it uses $\log Q^\pi(s,a)$ instead of $Q^\pi(s,a)$ and adds an extra term that depends on directly finding the gradient of the logarithmic action-value function (similar to DPG [10] that requires $\nabla_\theta Q(s, \pi_\theta(s))$. For discrete actions, here we would require a relaxation of the discrete action (mean estimate of the actions at a given state).

4 Off-Policy Policy Gradient Theorem

We can derive a similar policy gradient for off-policy where sampled experiences are under a behaviour policy $\mu(a|s)$. We start with the off-policy policy gradient objective $J(\pi) = \sum_s d_\mu(s) V^\pi(s)$. Importance sampling is one of the popular approaches in off-policy reinforcement learning [2]. Considering policy gradient with end of the trajectory returns, as in REINFORCE [11], where actions are sampled under the behaviour policy $\mu(a|s)$, the importance sampled policy gradient is given by $\nabla_\theta J(\theta) = (\prod_{t=1}^{T} \rho_t) \sum_{t=0}^{k} (\sum_{i=0}^{k} \gamma^i r_{t+i}) \nabla_\theta \log \pi_\theta(a_t|s_t)$, where ρ_t denotes the importance weighting. Equation **??** provides an unbiased gradient estimator, but it can suffer from high variance due to the product of unbounded importance sampling weightings. Off-PAC [7] attacked this problem by using the marginal value functions, and replaces the product

[1] Note that for stability purposes, we might require an empirical trick and consider a lower bound on the values in the logarithmic space by considering mappings with a δ of the form $f(x) = c\ln(x+\delta)$

of importance weights (required with an estimate of the marginal importance weights. Recent work, as in ACER [12], derives from Off-PAC for an off-policy policy gradient algorithm, but requires clipping of importance weights, while the critic is evaluated with λ-returns as in Retrace(λ) [13]. We start with the off-policy counterpart of the policy gradient objective, with marginal importnace weightings

$$\begin{aligned} J(\pi) &= \sum_s d_\mu(s) V^\pi(s) = \sum_s d_\mu(s) \sum_a \pi(a|s) Q^\pi(s,a) = \sum_s d_\mu(s) \sum_a \pi(a|s) \frac{\mu(a|s)}{\mu(a|s)} Q^\pi(s,a) \\ &= \mathbb{E}_{s\sim d_\mu(s), a\sim\mu(a|s)}[\frac{\pi(a|s)}{\mu(a|s)} Q^\pi(s,a)] \end{aligned} \tag{7}$$

Instead of using marginal importance weights, we argue that we can instead apply Jensen's inequality in equation 7. Jensen's inequality states that when f is concave, then $f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$. Applying Jensen's, and considering logarithmic mappings to equation 7, we therefore get a lower bound to the above policy gradient objective.

$$\begin{aligned} \tilde{J}(\pi) &= \log[\mathbb{E}_{s\sim d_\mu(s), a\sim\mu(a|s)}[\frac{\pi(a|s)}{\mu(a|s)} Q^\pi(s,a)] \\ &\geq \mathbb{E}_{s\sim d_\mu(s), a\sim\mu(a|s)}[\log[\frac{\pi(a|s)}{\mu(a|s)} Q^\pi(s,a)]] = \mathbb{E}_{s\sim d_\mu(s), a\sim\mu(a|s)}[\log \pi(a|s) + \log Q^\pi(s,a) - \log \mu(a|s)] \end{aligned} \tag{8}$$

Considering parameterized policies $\pi_\theta(a|s)$, the gradient of the lower bound to the objective is therefore

$$\begin{aligned} \tilde{J}(\theta) &= \mathbb{E}_{s\sim d_\mu(s), a\sim\mu(a|s)}[\nabla_\theta \log \pi_\theta(a|s) + \nabla_\theta \log Q^\pi(s,a) - \nabla_\theta \log \mu(a|s)] \\ &= \mathbb{E}_{s\sim d_\mu(s), a\sim\mu(a|s)}[\nabla_\theta \log \pi_\theta(a|s) + \nabla_\theta \log Q^\pi(s,a)] \end{aligned} \tag{9}$$

Equation 9 gives the off-policy policy gradient theorem where samples are under the behaviour policy $\mu(a,s)$ instead of $\pi(a,s)$. More interestingly, equation 9 has the same form as we previously showed with the policy gradient theorem with logarithmic mappings in equation **??** with the only difference of not having to sum over actions and having $\mathbb{E}_\mu$ instead of $\mathbb{E}_\pi$. This is an interesting result since the same form of the policy gradient theorem holds, for both the on-policy case (where $\mathbb{E}$ is under the target policy π), and the off-policy case (where $\mathbb{E}$ is under the target policy μ), requiring only minor changes. In the off-policy case, however, we would require the critic to be evaluated under off-policy samples with regular importance sampling corrections. We demonstrate, in the next section, how considering logarithmic mappings of the value function, we can also derive an off-policy critic in the logarithmic space.

5 Off-Policy Actor-Critic with Logarithmic Policy Evaluation

In this section, we derive an off-policy actor-critic algorithm following the policy gradient theorem in equation **??** and the off-policy counterpart of it in equation 9. In off-policy actor-critic, we can approximate the critic $\log Q^\pi(s,a) \approx \log Q^w(s,a)$ with a function approximator, and would need off-policy corrections for the critic evaluation, since we want to estimate $\log Q^\pi(s,a)$ while our samples are under the behaviour policy $\mu(a,s)$. We consider the one-step off-policy critic evaluation here, ie, off-policy SARSA or Expected SARSA, instead of the more popular variants with λ-returns as in Retrace(λ) [13].

However, note that, unlike the regular critic $Q^\pi(s,a)$ as in most actor-critic algorithms, we now have a logarithmic mapping of the critic evaluation $\log Q^\pi(s,a)$. In this section for the off-policy critic evaluation, we consider the value function mapped to the logarithmic space and perform the value function updates directly in this space. We emphasize that considering logarithmic mappings have an interesting outcome - for off-policy corrections, we can now directly avoid the importance sampling ratio.

We show the off-policy critic update directly in the logarithmic space as follows. Let us first consider the off-policy TD update, with an importance sampled correction given by $Q(s,a) := Q(s,a) + \alpha \frac{\pi(a|s)}{\mu(a|s)}[r(s,a) + \gamma Q(s',a') - Q(s,a)]$.

Recent work (under submission) shows the existence of a non-linear Bellman equation with logarithmic mappings of the value function. Considering the off-policy update with logarithmic value functions, we can write the update as

$$\tilde{Q}(s,a) := \tilde{Q}(s,a) + \alpha f[\frac{\pi(a|s)}{\mu(a|s)}[r(s,a) + \gamma f^{-1} Q(s',a') - Q(s,a)]] = \tilde{Q}(s,a) + \alpha f[\frac{\pi(a|s)}{\mu(a|s)}] + \alpha f[r(s,a) + \gamma f^{-1} \tilde{Q}(s',a') - \tilde{Q}(s,a)] \tag{10}$$

and since $f(x) = \log(x)$, we can write the above off-policy update as $\tilde{Q}(s,a) = \tilde{Q}(s,a) + \alpha[\log \pi(a|s) - \log \mu(a|s)] + \alpha f[r(s,a) + \gamma f^{-1}\tilde{Q}(s',a') - \tilde{Q}(s,a)]$. Alternately, we can also consider Expected SARSA off-policy update for the critic evaluation, in the logarithmic space, such that $\tilde{Q}(s,a) := \tilde{Q}(s,a) + \alpha f[r(s,a) + \gamma \sum_a \pi(a|s) f^{-1}(\tilde{Q}(s,a)) - \tilde{Q}(s,a)]$.

We can similar minimize the MSE loss for the critic update, considering parameterized logarithmic critics $\tilde{Q}^w(s,a)$ where the MSE with an Expected SARSA update is given by $L(w) = \mathbb{E}_{s\sim d_\mu(s), a\sim\mu(a|s)}[(y - \tilde{Q}(s,a))]$, where the target is given by $y = \log[r(s,a) + \gamma \sum_{a'} \pi(a'|s') \exp \tilde{Q}(s',a')]$. Following our off-policy gradient theorem, as in equation 9, and considering logarithmic mappings of the off-policy critic evaluation, we can therefore derive an off-policy actor critic algorithm

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d_\mu, a\sim\mu(a|s)}[\nabla_\theta \log \pi_\theta(a|s) + \nabla_\theta \log Q^w(s,a)] \tag{11}$$

where in the actor updates, we update θ following the gradient update in equation 11, and the logarithmic critic approximation with parameters w is a one step TD update with Expected SARSA, or any off-policy policy evaluation algorithms.

6 Conclusion and Future Work

In this work, we provide a derivation of the off-policy policy gradient theorem, which relies on logarithmic mappings of the value functions. We show that a particular interesting property considering non-linear value functions is that, the resulting policy gradient theorem, and the off-policy counterpart, has the similar form of the gradient such that we can either compute the gradients on-policy or with behaviour samples in off-policy learning. As a result, we can conveniently avoid marginal importance sampling corrections in our resulting off-policy gradient theorem.

We have provided a derivation of our off-policy policy gradient theorem. In future work, it would be interesting to see whether this form of the gradient is useful in both on-policy or off-policy case. Particularly in off-policy learning, if this approach provides a way to completely avoid importance sampling corrections, it would be interesting to analyse the effectiveness of this derivation, compared to the other high variance counterparts of off-policy policy gradient methods.

References

[1] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 2000.

[2] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[3] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1889–1897, 2015.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[5] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[6] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. *CoRR*, abs/1806.06920, 2018.

[7] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[8] Hamid Reza Maei. Convergent actor-critic algorithms under off-policy training and function approximation. *CoRR*, abs/1802.07842, 2018.

[9] Ehsan Imani, Eric Graves, and Martha White. An off-policy policy gradient theorem using emphatic weightings. In *Advances in Neural Information Processing Systems*, 2018.

[10] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

[11] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

[12] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *International Conference on Learning Representations*, 2017.

[13] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1046–1054, 2016.

Predicting Human Choice in a Multi-Dimensional N-Armed Bandit Task Using Actor-Critic Feature Reinforcement Learning

Tyler James Malloy
Department of Cognitive Science
Rensselaer Polytechnic Institute
Troy, NY 12180
mallot@rpi.edu

Rachel A. Lerch
Department of Cognitive Science
Rensselaer Polytechnic Institute
Troy, NY 12180
lerchr2@rpi.edu

Zeming Fang
Department of Cognitive Science
Rensselaer Polytechnic Institute
Troy, NY 12180
fangz5@rpi.edu

Chris R. Sims
Department of Cognitive Science
Rensselaer Polytechnic Institute
Troy, NY 12180
simsc3@rpi.edu

Abstract

Recent improvements in Reinforcement Learning (RL) have looked to integrate domain specific knowledge into determining the optimal actions to make in a certain environment. One such method, Feature Reinforcement Learning (FRL) [1], alters the traditional RL approach by training agents to approximate the expected reward associated with a feature of a state, rather than the state itself. This requires the value of a state to be some known function of the values of the state features, but it can accelerate learning by applying experience in one state to other states that have features in common. One domain where these assumptions hold is the multi-dimensional n-armed bandit task, in which participants determine which feature is most associated with a reward by selecting choices that contain those features. In this environment, the expected reward associated with one choice is the sum of the expected reward associated with each of its features. The FRL approach displayed improved performance over traditional q-learning in predicting human decision making. Similar improvements in the speed of learning have been displayed in some environments by using an actor-critic (AC) model, which uses a second-order learning strategy where the state-value function $v(s)$ can be considered as a critic of the state-action-value function q(s,a) [2]. In this paper we apply the domain specific knowledge approach of FRL onto AC to develop an Actor-Critic Feature RL (AC-FRL) model and display improved performance over FRL in predicting human choice decisions in the multi-dimensional n-armed bandit task. These improvements in performance are most closely connected to the increased confidence that the AC-FRL model has in it's predictions, particularly in games where the participant may have not learned which feature was most associated with a reward.

Keywords: Reinforcement Learning, Actor-Critic, Multi-armed Bandits

Acknowledgements

This research was supported by NSF grant DRL-1560829 and a grant from the RPI-IBM Artificial Intelligence Research Collaboration (AIRC). Empirical data was provided by Yael Niv from the Niv Lab Website.

1 Introduction

Modelling human learning using Reinforcement Learning (RL) entails addressing many issues and open questions. Among the most prominent is how individuals learn generalizable policies in high dimensional domains, or resolve the so-called 'curse of dimensionality'. One method of mitigating this issue is to implement domain specific algorithms that use some predefined knowledge of the environment in their learning strategy. Although this method makes these models less flexible (and imposes 'inductive bias'), it ideally reflects the types of assumptions humans also make while trying to improve their decision making in complex tasks. To study this issue, Niv et al. [1] developed a small-scale laboratory task, in the form of a multi-dimensional n-armed bandit task, that nonetheless captures many of the important issues involved in learning in high-dimensional environments. This task consists of human participants selecting options with different combinations of randomly generated features, and determining which single feature was most associated with a reward.

To tackle the issues with predicting human decision making in complex environments, Niv et al. developed a 'Feature Reinforcement Learning' (FRL) model, based on learning a state-action value function, $q(s, a)$, but integrated with domain specific knowledge of the task, namely that the value of an option should be a weighted (linear) combination of the features that make up that option. However, this feature representation should be applicable to any RL learning method, and one interesting candidate is the actor-critic (AC) learning architecture. One feature of AC that make it a good candidate for this specific environment is that it may potentially better reflect the neural architecture of human reinforcement learning [2]. In this paper we apply the same feature representation previously used by the FRL model, but within the context of an actor-critic RL model. Using this Actor-Critic Feature Reinforcement Learning (AC-FRL) method, we show improved accuracy in predicting human choice in the multi-dimensional n-armed bandit task.

2 Background

2.1 Multi Dimensional N-Armed Bandit Task

In this task, participants were presented on each trial with a choice of three objects, each composed of three unique features, and were required to determine which feature among 9 possibilities was associated with an increased probability of observing a reward (75% vs 25%). The participants played games lasting uniformly between 15-25 trials in which the target feature was the same, and all 9 features were present in each trial. The data analyzed in this paper consisted of 22 participants each with 800 trials, with 500 fast-paced (500 ms) and 300 slower-paced trials. The possible features were the color (red, blue, green), shape (square, triangle, circle), and texture (dotted, hatch, wavy). Each trial contained all 9 features in different combinations determined randomly in 3 choices. A more complete methodology is presented in [1].

2.2 Feature Reinforcement Learning

In the Reinforcement Learning (RL) setting, an agent's goal is to learn (or approximate) an optimal policy function $\pi^\star(a|s)$ which gives a probability distribution over possible actions for a given state. A fundamental construct in RL to accomplish this is an optimal state-action function, $q^*(s, a)$, that returns the expected long-term reward associated with performing action in a state, and subsequently following an optimal policy. The Feature Reinforcement Learning (FRL) algorithm presented in [1] facilitates learning in multidimensional environments by decomposing the state-action value function using a linear combination of features:

$$q_\omega(s, a) = \omega^T \Phi(s_a) \tag{1}$$

where the state s consists of a collection of three objects presented to the subject on a given trial, the action a corresponds to selecting one of the objects, and s_a is the selected object. $\Phi(s_a)$ returns the feature vector associated with that object, corresponding to the features that make up any stimulus such as [red, yellow, green, triangle, square, circle, dotted, wavy, hatch]. For example the vector returned by $\Phi(s_1)$ where the action is selecting a red-dotted-triangle would be $[1, 0, 0, 1, 0, 0, 1, 0, 0]$. The state-action value $q_\omega(s, a)$ is thus a weighted combination of the features that compose the chosen object, where the vector ω defines the feature weights learned using standard TD-learning with linear function approximation [2]. In particular, given an observed reward r and learning rate α, the learning rule is:

$$\omega \leftarrow \omega + \alpha[r - \omega^T \Phi(s_a)]\Phi(s_a) \tag{2}$$

Given state-action values for each choice, $q_\omega(s, a)$, the FRL algorithm's choice policy is defined by a soft-max distribution over these values.

The underlying feature representation reduces the number of trials required to approximate the value of a state by generalizing the information gathered from an outcome beyond the state it took place in. In the next section we detail the application of this same approach to a model based on actor-critic learning.

2.3 Actor-Critic Feature Reinforcement Learning

The actor-critic architecture decomposes the basic RL problem into separate sub-problems for learning a value function, and learning a policy. Notably, this introduces a form of second-order learning dynamics, meaning that it is incrementally

updating two interacting learning components. The first component is the state-action function $q(s, a)$ which approximates the expected long term reward of performing an action in a state for the current policy. The second component is the policy function $\pi(a|s)$ which is itself adapting to changes in $q(s, a)$ in order to improve behavior, and maps the state onto the probability of performing any action in that state. In contrast, other RL algorithms such as SARSA or q-learning lack this second-order learning property, as the policy is directly and immediately determined by the current value function. The AC-FRL model uses the same linear function approximation $q_\omega(s, a)$ as the FRL model. AC-FRL differs in that the policy $\pi_\theta(a|s)$ is also parameterized in terms of a linear combination of features, via the parameter vector θ:

$$\pi_\theta(a|s) = \frac{\exp[\beta\, \theta^T \Phi(s_a)]}{\sum_{a'} \exp[\beta\, \theta^T \Phi(s_{a'})]}. \tag{3}$$

This policy is updated in our model by minimizing the following objective function:

$$J_\theta = \sum_a \pi_\theta(a|s)(\max_{a'} q(s, a') - q(s, a)). \tag{4}$$

Note the similarity between the target of this update rule, $(\max_{a'} q(s, a') - q(s, a))$, and the so-called "advantage function" [3], defined as $A(s, a) = q(s, a) - v(s)$. Maximizing the advantage function is equivalent to minimizing the objective function given in Eq. 4. This objective penalizes actions that deviate from the best possible action, given the current estimate for $q(s, a)$. Following the gradient of this objective function gives the learning rule for the state-value-function parameterization vector as $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta J_\theta(s)$. For the linear feature representation used in our model, this yields

$$\theta \leftarrow \theta + \alpha_\theta[r - (\max_a q(s, a') - q(s, a))]\Phi(s_a). \tag{5}$$

We parameterize $q(s, a)$ with the same feature representation used in the FRL model: $q_\omega(s, a) = \omega^T \Phi(s, a)$, and utilize the same update rule (Eq 2).

The mapping of a state-action pair onto the $\Phi(s_a)$ vector includes the domain specific knowledge that the value of a choice is the sum of the features it consists of. Multiplying by $\Phi(s_a)$ ensures that only the values that correspond to the features that are make up the selected option are updated based on the observed reward. The result of this update policy is that, at each time step, the 3 features that make up the choice that was selected by the agent have their corresponding ω and θ values updated by the value that the participant observed after selecting that choice.

3 Results

Model comparison was performed in the same manner as described in the Niv paper [1] by using leave-one-out cross-validation. For each game, model parameters were determined by minimizing the negative log-posterior of the participant's data in relation to these parameters while excluding the given game. Then the model with those parameters was used to determine the likelihood of the choices in the left out game. Results from this process are plotted on Figure 1. This process results in a more valid model comparison by avoiding biases introduced by over-fitting model parameters to a given set of participant choices. Results from this comparison show that the performance of the Actor-Critic version of the model outperform the average likelihood of standard Feature RL. Results are split between learned and unlearned games. A game was considered to be learned when the participant correctly selected the option containing the feature of interest on 4 or more of the last 6 trials within the game.

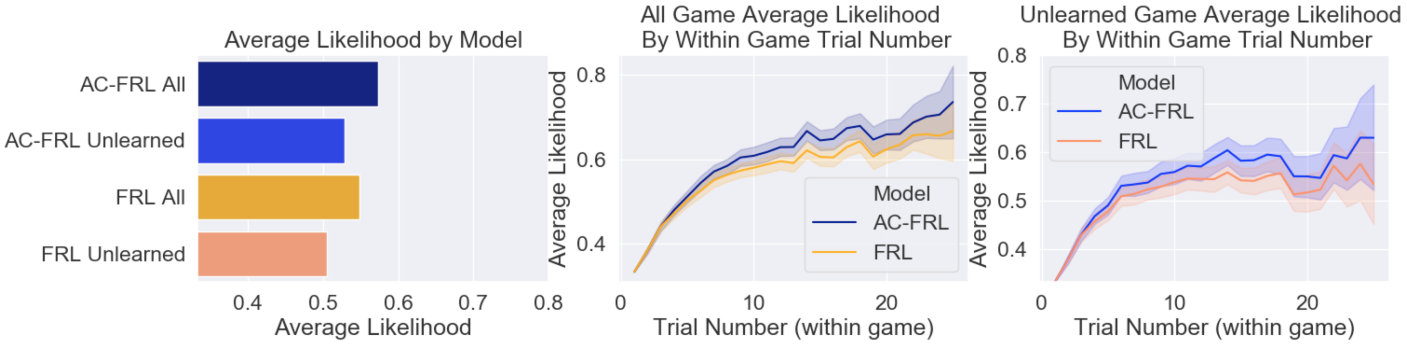


Figure 1: Average likelihood assigned by each model to the choice that the participant selected by within game trial number. The Actor-Critic model predicted participants' performance significantly better ($p < 0.05$) than the standard Feature RL method for trials 5-20. Transparent error bars signify 95% confidence interval.

3.1 Average Likelihood by Trial Number

Figure 1 shows improves performance by the Actor-Critic version of the Feature RL model [1] in predicting human choice selection for the multi-dimensional n-armed bandit task environment. Both the Actor-Critic and standard models incorporate parameters for the learning rate (AC-FRL uses 2 learning rates for the ω and θ update rules) and β soft-max inverse temperature. A gamma distribution prior with size 2 and shape 3 was used for the β while fitting these parameters for both models. The Actor-Critic model had an AIC averaged by games of 30.39, Feature RL model had 32.37.

3.2 Model Prediction Confidence

An alternative to considering the average likelihood is to allow the models to make a prediction based on their policy and evaluate the probability that the model assigns to its selection. Figure 2 shows the average probability that the model assigned to the prediction it made in this scenario. These results show that the Actor-Critic method has higher confidence in its predictions, especially for unlearned games, than Feature RL. Critically the Actor-Critic method matches the early trial confidence of traditional Feature RL, meaning it does not simply identify the feature of interest faster and with a higher confidence. Since the performance of the participants in unlearned games suggests they never learned what the feature of interest was, it is likely that many of these trials included instances where the participant was changing their belief on what the feature of interest was. These unlearned game predictions are difficult because the participant may still be exploring different options as to what they believe the feature of interest is, so a high confidence and accuracy for these trials is a key factor in the improved predictive accuracy of the Actor-Critic model. Although it is difficult to make general statements on the performance of different RL methods, the results discussed in this section suggest that part of the improved performance of Actor-Critic comes from its faster and more flexible learning.

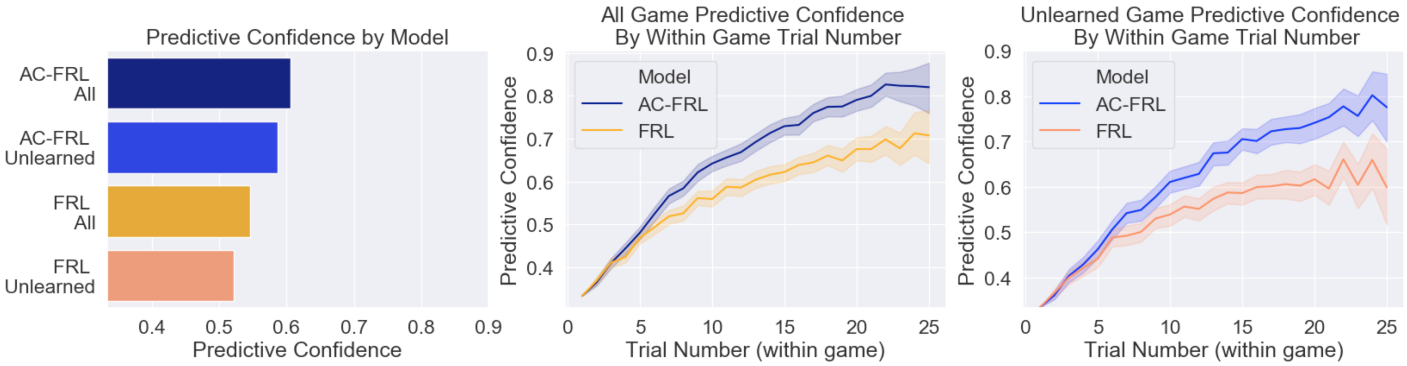


Figure 2: Confidence assigned to the choice that was ultimately selected by the model by within game trial number. Transparent error bars signify 95% confidence interval.

3.3 Simulated Game Environment

To better understand the improvement in average likelihood gained by the Actor-Critic version, we developed a simulated game environment where the two methods engaged in the same basic task as the human participants, but where we altered the reward probability associated with choosing the correct feature. We investigated simulated environments with 50%, 75% and 100% probabilities of observing a reward after selecting the option with the correct response P_r, and kept the probability of observing a reward after an incorrect selection to be (1- P_r). The only significant difference in probability of selecting the feature of interest between the FRL and AC-FRL algorithms in a simulated game environment occurs when the probability of viewing a reward after selecting the option with the correct feature is 100% with a 0% chance of receiving a reward after an incorrect guess.

Method	50% Reward Probability	75% Reward Probability	100% Reward Probability
AC-FRL Feature	44 %	60%	79%
FRL Feature	45%	63%	84%

Firstly, these results indicate that the improvement in average likelihood by the AC-FRL algorithm is not merely because the algorithm is learning to play the game better than the FRL model. This was an important finding, as the higher confidence that the AC-FRL model assigns to the correct choice could potentially have been the result of the model being better at playing the game in a way that doesn't reflect human learning strategies. In fact, The accuracy in selecting the choice that contained the feature of interest of our model with the base rate probabilities for reward from correct/incorrect

[1]Niv et al [1] reported improved performance by adding a parameter that decays the values of non-chosen features towards zero. Adding this parameter to AC-FRL also showed similar performance improvements and does not change the interpretation of our results. Hence for simplicity we omit it from the current discussion.

guesses (75% and 25%) shows a slight decrease in performance compared to FRL. Secondly, this suggests that one possible explanation for the improvement in average likelihood of the Actor-Critic model is in part due to instances where the participant observed no reward after selecting the option with the correct feature. Since the standard Feature RL model performed better when the reward probability was deterministic, it may be the case that the Actor-Critic model more accurately reflects the changes in approximate value that the participants encoded as a result of observing no reward when selecting the an option with the feature of interest, or vise versa.

4 Conclusions

The AC-FRL model described in this paper displayed improved performance in predicting human decision making in the multi-dimensional n-armed bandit environment. The source of this improved performance was partially due to the increased confidence in predictions, particularly in unlearned games when the participant may not have learned the feature of interest. This increased confidence was a result of faster and more flexible learning afforded by using the two-tiered architecture implicit in the Actor-Critic model. The current work is relevant to any attempt at improving performance by applying principled domain specific knowledge of a task onto a traditional learning model. Additionally, there has been some recent interest in the use of the advantage function described in [3], which shows a close connection to the update rule used in the θ parameter update method.

Attention is a valuable and limited resource for all intelligent agents, meaning one key aspect to acting in high-dimensional environments is the selection of a subset of relevant features to attend to. The FRL method approaches the issue of attention by directing the agent to approximate the values of features that occur repeatedly across each state, and thus inform the value of all states. While the original FRL model displayed high accuracy in predicting human choice, this general method can be applied to other RL techniques outside of a traditional q-learning strategy. The AC model is a good candidate for applications of domain specific knowledge due to the fact that both are attempting to improve performance by learning from fewer training examples. This quality of the AC model is particularly relevant to predicting human selection in the multi-dimensional n-armed bandit task. This is because the number of training examples is far fewer than the state space, and the human participants are naturally superior to traditional methods at learning in these high dimensional environments from few examples. The results presented in this paper displayed the ability of a feature based approach to be applied onto the actor-critic architecture.

References

[1] Y. Niv, R. Daniel, A. Geana, S. J. Gershman, Y. C. Leong, A. Radulescu, and R. C. Wilson. Reinforcement Learning in Multidimensional Environments Relies on Attention Mechanisms. *Journal of Neurosci*, 35(21):8145–8157, 2015. 1, 2, 3

[2] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 1

[3] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000. 2, 4

5 Supplementary Material

Algorithm 1 Feature based Capacity-Limited Actor-Critic for Multi-Dimesnional N-Armed Bandit Task

Input: Trial stimuli: s
Input: Participant choices: a
Input: Trial observed rewards: r
Parameters: Learning rates $0 \leq \alpha_\omega, \alpha_\theta \leq 1$, Soft-max inverse temperature $\beta > 0$.
Initialize policy parameterization: $\omega \in \mathbb{R}^d$ and state-action value parameterization $\theta \in \mathbb{R}^d$
for *each trial (for each participant):* **do**
    Compute the choice probabilities for the current trial:
    $\pi_\theta(a|s) \leftarrow \dfrac{\exp[\beta\theta^T\Phi(s_a)]}{\sum_{i=1}^{3}\exp[\beta\theta^T\Phi(s_i)]}$
    Predict action $a \sim \pi_\theta(a \mid s)$
    Update based on the human choice and observed reward:
    $\omega \leftarrow \omega + \alpha_\omega[r - \omega^T\Phi(s_a)]\Phi(s_a)$
    $q(s,a) \leftarrow \omega^T\Phi(s_a)$
    $\theta \leftarrow \theta + \alpha_\theta[r - (\max_{a'} q(s,a') - q(s,a))]\Phi(s_a)$
end for

A Value Function Basis for Nexting and Multi-step Prediction

Andrew Jacobsen **Vincent Liu** **Roshan Shariff** **Adam White** **Martha White**

Department of Computing Science
University of Alberta
Edmonton, AB T6G 2E8
{ajjacobs, vliu1, roshan.shariff, amw8, whitem}@ualberta.ca

Abstract

Humans and animals continuously make short-term cumulative predictions about their sensory-input stream, an ability referred to by psychologists as *nexting*. This ability has been recreated in a mobile robot by learning thousands of value function predictions in parallel. In practice, however, there are limitations on the number of things that an autonomous agent can learned. In this paper, we investigate inferring new predictions from a minimal set of learned General Value Functions. We show that linearly weighting such a collection of value function predictions enables us to make accurate multi-step predictions, and provide a closed-form solution to estimate this linear weighting. Similarly, we provide a closed-form solution to estimate value functions with arbitrary discount parameters γ.

Keywords: Reinforcement Learning, Multi-Step Prediction

1 Introduction

The ability to continually make predictions about one's sensory-motor stream is an important aspect of forming awareness of one's environment. In particular, it has been shown that both humans and animals continually make large numbers of short-term cumulative predictions about their sensory input at many different time-scales [Fedus et al., 2019, Pezzulo, 2008, Carlsson et al., 2000, Brogden, 1939]. This ability is referred to as *nexting*. Recent work in Reinforcement Learning has been able to recreate this ability in a mobile robot by using a collection of *General Value Functions* (GVFs)[Sutton et al., 2011], learned online and in parallel [Modayil et al., 2014]. GVFs make cumulative predictions about a given target; in a standard value function, for example, this target is the reward signal. When we instead let the target be an observation from the sensory-motor stream, the GVF prediction corresponds to a nexting prediction.

There are are limitations, however, on the number of predictions that an agent can make in a continual learning setting. Each nexting prediction that the agent makes has its own cost in terms of memory and computation. With a large enough collection of nexting predictions — say in the millions — it becomes infeasible for the agent to be able to update them all. Furthermore, in this setting we want our agents to be able to make *new* predictions during run-time. This can be problematic since each GVF may require different hyperparameters (learning rate α, trace decay rate λ, etc.) to learn accurate predictions. Because of this, not only does the agent have to learn each new prediction from scratch, but may have to do so multiple times in order to find the right hyperparameters — all before being able to actually use that prediction.

Instead of learning all possible predictions from scratch, in this paper we investigate whether an agent can use a small set of sufficiently informative nexting predictions to infer the answers to other questions. We show that a small collection of GVF predictions can be used to accurately estimate the answers to predictive questions that the agent has not explicitly learned. We introduce a simple linear transformation which uses a collection of GVF predictions to estimate 1) other GVFs with arbitrary discounting parameters, and 2) n-horizon predictions.

This work has a similar motivation to multi-scale Successor Representations (SRs) [Momennejad and Howard, 2018], and Universal Value Function Approximators (UVFAs)[Schaul et al., 2015]. SRs, in fact, can be represented as GVFs. This work differs from multi-scale SRs, because they assume a tabular setting and use a different weighting scheme with approximate laplace transforms. UVFAs focus on learning value functions that generalize across goal states, using neural networks.

2 General Value Functions

In this section, we define the concepts of return and value and their extension to more general predictions. Consider a Markov Reward Process defined by state-space $\mathcal{S}$, transition function $P : \mathcal{S} \times \mathcal{S} \mapsto [0, 1]$, and reward function $r : \mathcal{S} \mapsto \mathbb{R}$ defined as $r(s) = \mathbb{E}\big[R_{t+1}|S_t = s\big]$, where R_t and S_t are random variables representing the reward and state at time t respectively. We define the return at time t to be

$$G_t \coloneqq \sum_{j=0}^{\infty} \gamma^j r(S_{t+j})$$

where $\gamma \in [0, 1)$ is a constant discounting factor. Given a state $s \in \mathcal{S}$, we define the *value* of state s to be the expected return from state s

$$v(s) \coloneqq \mathbb{E}\big[G_t|S_t = s\big]$$

The function $v(s)$ is referred to as the *Value Function*. A *General Value Function* (GVF) [Sutton et al., 2011] extends the above definition of value, by allowing $r(S_t)$ to be *any* function of the current state — not just a reward signal — and letting the discounting factor be a function of state as well, $\gamma_t \coloneqq \gamma(S_t)$. In this paper we consider only constant discounting factors, thus the above definition of return does not change. GVF predictions with $r(\cdot)$ set to the observations correspond to nexting predictions [Modayil et al., 2014].

3 Predicting Future Outcomes with General Value Functions

In this section, we explain how a set of value function predictions can be used to approximate outcomes n-steps into the future. We start by assuming access to the actual returns into the future and then discuss implications when using expected returns and estimation with value functions.

Suppose we are interested in reconstructing an unknown time series $y_1, \ldots, y_t, \ldots \in \mathbb{R}$. Suppose further that we know the discounted sum of this time series, for several discounts $\gamma_1, \ldots, \gamma_k \in [0, 1)$:

$$G_{t,\gamma_i} = \sum_{j=0}^{\infty} \gamma_i^j y_{t+j+1}.$$

Our goal is to reconstruct various aspects of y given only $G_{t,\gamma_1}, \ldots, G_{t,\gamma_k}$. We are primarily interested in reconstructing y_n for a variety of horizons $n \in \mathbb{N}$. We might also be interested in reconstructing G_γ for some $\gamma \notin \{\gamma_1, \ldots, \gamma_k\}$.

In general, obtaining exact reconstructions is not possible, because we only have k known quantities and yet we want to reconstruct y_n for all $n \in \mathbb{N}$. Our only recourse is to *approximate* y. To do so, define the function $f : \mathbb{N} \to \mathbb{R}$, with $f(t) \coloneqq y_t$. We will try to find a $\hat{f}$ that minimizes the distance to f.

To begin with, we can think of f as an element of an infinite-dimensional vector space — the space of all functions $\mathbb{N} \to \mathbb{R}$. We can define an inner product on this space: $\langle f, g \rangle = \sum_{t=0}^{\infty} f(t)g(t)$, for $f, g \in \mathbb{N} \to \mathbb{R}$, which in turn gives us a norm $\|f\| = \sqrt{\langle f, f \rangle} = \sqrt{\sum_{t=0}^{\infty} f(t)^2}$. An element of this function space is the function $t \mapsto \gamma^t$. We denote this function $\vec{\gamma}$ to distinguish it from the scalar γ. We can see that the discounted sum is actually an inner product: $G_\gamma = \langle \vec{\gamma}, f \rangle$. In other words, we have a system of k linear equations in the unknown f:

$$\begin{aligned} \langle \vec{\gamma}_1, f \rangle &= G_{t,\gamma_1} \\ &\vdots \\ \langle \vec{\gamma}_k, f \rangle &= G_{t,\gamma_k} \end{aligned}$$

As mentioned above, it is impossible to recover f from this system since f is infinite-dimensional and the dynamics of $f(t)$ are unknown. However, we *do* have k infinite-dimensional known quantities: the functions $\vec{\gamma}_i$. Therefore, we can at least recover the component of f that lies in the k-dimensional subspace spanned by $\vec{\gamma}_1, \ldots, \vec{\gamma}_k$. Define

$$\hat{f}_{\boldsymbol{\theta}}(t) \coloneqq \sum_{i=1}^{k} \theta_i \gamma_i^t$$

for $\boldsymbol{\theta} \in \mathbb{R}^k$ as a linear combination of the functions $\{\vec{\gamma}_1, \ldots, \vec{\gamma}_k\}$ with coefficients $\theta_1, \ldots, \theta_k$. We want to find the coefficients $\boldsymbol{\theta}$ that minimize the squared distance $\|\hat{f}_{\boldsymbol{\theta}} - f\|^2$. Let G be the matrix with the infinite-dimensional $\vec{\gamma}_i$ as its columns. Then the $\boldsymbol{\theta}$ which minimizes $\|\hat{f}_{\boldsymbol{\theta}} - f\|^2$ is the least squares solution to $G\boldsymbol{\theta} \approx f$. That is,

$$G^\top G\boldsymbol{\theta} = G^\top f \implies \boldsymbol{\theta} = (G^\top G)^{-1} G^\top f = K^{-1} \begin{bmatrix} \langle \vec{\gamma}_1, f \rangle \\ \vdots \\ \langle \vec{\gamma}_k, f \rangle \end{bmatrix}, \tag{1}$$

where K is a $k \times k$ matrix whose entries are given by $K_{ij} = \langle \vec{\gamma}_i, \vec{\gamma}_j \rangle$. Fortunately, the entries of K can actually be computed in closed form when the discounting factors are constant:

$$K_{ij} = \langle \vec{\gamma}_i, \vec{\gamma}_j \rangle = \sum_{t=0}^{\infty} \gamma_i^t \gamma_j^t = \frac{1}{1 - \gamma_i \gamma_j}$$

for $0 \leq \gamma_i, \gamma_j < 1$. Note that certain selections of discounting factors can lead to a poorly conditioned K matrix; the conditioning can be improved by applying ℓ_2 regularization in the usual way, by replacing K with $K + \lambda I$ in Equation 1, where λ is the weight of the regularization and I is a $k \times k$ identity matrix.

With this approximation to f, we can return to the problem of approximating aspects of the series y. We can obtain n-horizon predictions by using

$$y_n = f(n) \approx \hat{f}_{\boldsymbol{\theta}}(n) = \sum_{i=1}^{k} \theta_i \gamma_i^n.$$

We can estimate the discounted sum $\langle \vec{\gamma}, f \rangle$ for some $\gamma \notin \{\gamma_1, \ldots, \gamma_k\}$ using

$$\sum_{i=0}^{\infty} \gamma^t y_{t+1} = \langle \vec{\gamma}, f \rangle \approx \langle \vec{\gamma}, \hat{f}_{\boldsymbol{\theta}} \rangle = \sum_{i=1}^{k} \theta_i \langle \vec{\gamma}, \vec{\gamma}_i \rangle = \sum_{i=1}^{k} \frac{\theta_i}{1 - \gamma_i \gamma}. \tag{2}$$

When making predictions about the future, we do not have access to exact returns. Rather, we will estimate value functions — estimate expected returns — to use within the above formulas. For exact value functions, we can obtain the same approximations as above for expected n-step values and expected discounted sums. This is appropriate, as any direct multi-step prediction method using squared error is estimating the expected value n steps in the future. The approximation of the value functions themselves will introduce additional approximations to above.

4 Experimental Results

In this section we give two simple demonstrations of an agent's ability to infer the answers to questions it has not been trained to predict. We imagine a scenario in which the agent is performing a simple prediction task for a number of evaluation steps. Mid-way through the task, the agent adds a new prediction to be evaluated.

4.1 Predicting Future Observations

To demonstrate our method's ability to make n-horizon predictions, we tested our approach on the Mackey-Glass time series, a single-variable dataset derived from the time-delay differential equation:

$$\frac{\partial y(t)}{\partial t} = \alpha \frac{y(t-\tau)}{1+y(t-\tau)^{10}} - \beta y(t)$$

In this experiment, we used $\tau = 17$, $\alpha = 0.2$, and $\beta = 0.1$, starting from an initial value of $y(0) = 1.2$. The agent makes predictions at a horizon of 6 steps for $1,000,000$ steps, and adds a horizon 12 prediction mid-way through. We gave the agent a GVF basis consisting of 100 GVFs and constant discount factors $\gamma_i = 1.0 - i/101$, for $i = 1, \ldots, 100$. The GVFs were trained using TD(0) with linear value function approximation. The features given to the GVFs were a history of the previous 4 observations. We additionally included predictions made using a linear autoregressive model (*"Direct AR"*), with a history of 4 observations, as an optimal baseline. Note that the baseline was allowed to directly train on each of the horizons of interest, while our method was never explicitly trained to do any n-horizon prediction.

Results on this task are shown in figure 1 (Left). We can see that at the start of training, the GVF basis predictions at horizon 6 take a bit longer to learn, but still end up reaching the same performance level as the baseline without ever being trained to make this prediction. At the 500,000 step mark, the agent begins making horizon 12 predictions. Note that the baseline agent using direct AR has to wait almost 100,000 steps before it can obtain a reasonably accurate horizon 12 prediction! Furthermore, note that the GVF basis could have just as easily made predictions for an arbitrary number of horizons, all of which can be made immediately.

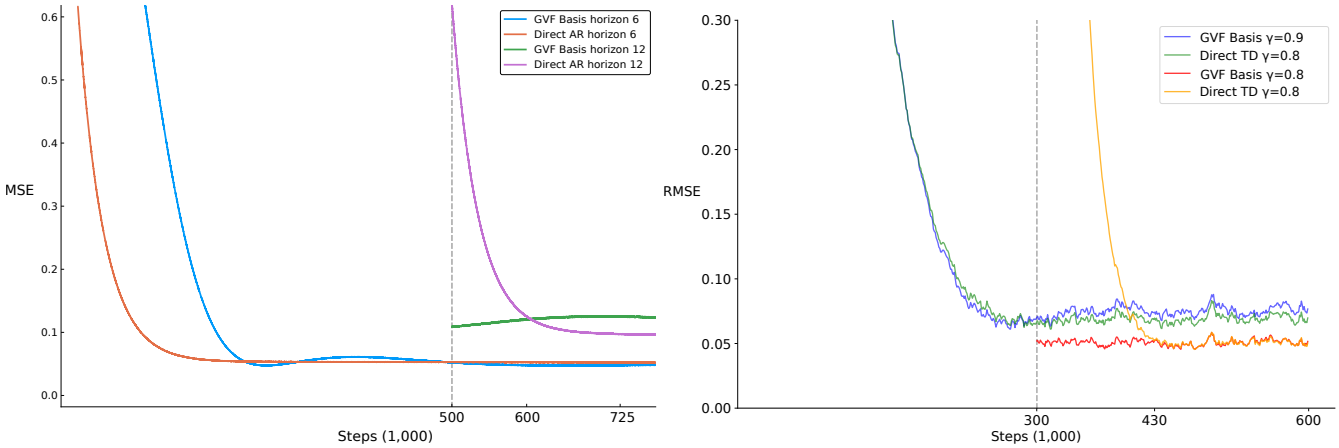


Figure 1: **Left:** Online Mean Squared Error (MSE) for the GVF Basis and Direct AR. For the first 500,000 steps of training, the agent makes only horizon 6 predictions. At step 500,000 the agent adds a horizon 12 prediction to evaluate. Performance stabilizes at around 725,000 steps, so we omit the final 250,000 steps. **Right:** Online Root Mean Squared Error (RMSE) for the GVF Basis and Direct TD.

Our method is slightly less accurate than the final performance of the baseline at horizon 12. However, we note that the discounting values are chosen rather arbitrarily; a better understanding of discount selection strategy could lead to improved performance. Selecting optimal discounting factors is still currently a work in progress at the time of writing. We note also that in these experiments, the GVFs were given a history of observations in order to build up sufficient state information. We could have instead used the GVF predictions themselves as features at each step, as in Schlegel et al. [2018]. We find that this approach works much better in general, but is outside the scope of this paper.

4.2 Predicting other General Value Functions

We also tested the accuracy of our method for predicting other GVFs. The environment is a randomly generated Markov chain with 500 states and the branching factor of 5 (the number of successor states), where we can compute the true value functions exactly. *GVF basis* learns the GVFs of a set discounted factors Γ_{train}, and predicts the GVF of a different discount factors γ. *Direct TD* learns the GVF of the discount factor γ directly.

Both methods use TD(0). We tuned the regularization constant over the values $\lambda \in \{0.0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and the learning rate over the set $\{0.4, 0.2, 0.1, 0.05, 0.025\}$. Online performance is shown in Figure 1 (Right). For the first 300,000 steps of training, the agent makes prediction for $\gamma_1 = 0.9$. We can see that the GVF Basis predictions are comparable to Direct TD predictions in the beginning of training. At step 300,000 the agent adds a prediction for $\gamma_2 = 0.8$. The GVF Basis can immediately make the new prediction accurately; the Direct TD method, however, needs to learn the value function from scratch, and takes roughly 430,000 steps to reach the same performance as the GVF Basis estimate.

5 Conclusions

In this work, we introduced a novel approach to infer new GVF predictions and multi-step predictions from a small set of learned GVFs. This work was focused more on whether a collection of GVF predictions *can* be used to make other predictions, rather than on the general utility of this approach. In these initial experiments, the collection of discounting factors Γ was chosen naively; future work will investigate how these discounting factors can be chosen optimally to facilitate reconstructing multi-step and discounted cumulative predictions. It is possible that better performance can be attained by selecting Γ in a more principled way. We note also that discounted cumulative predictions are interesting in their own right for time series prediction problems such as section 4.1; each of the predictions made with a different discounting factor provides slightly different information about how the signal is expected to evolve over time. We believe that the fact that predictions of this sort also facilitate relatively accurate multi-step predictions could make them a subject of interest to the general time series forecasting community.

References

Wilfred J Brogden. Sensory pre-conditioning. *Journal of Experimental Psychology*, 25(4):323, 1939.

Katrina Carlsson, Predrag Petrovic, Stefan Skare, Karl Magnus Petersson, and Martin Ingvar. Tickling expectations: neural processing in anticipation of a sensory stimulus. *Journal of cognitive neuroscience*, 12(4):691–703, 2000.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

Joseph Modayil, Adam White, and Richard S Sutton. Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior*, 22(2):146–160, 2014.

Ida Momennejad and Marc W. Howard. Predicting the future with multi-scale successor representations. *bioRxiv*, page 449470, October 2018. doi: 10.1101/449470. URL `https://www.biorxiv.org/content/10.1101/449470v1`.

Giovanni Pezzulo. Coordinating with the future: the anticipatory nature of representation. *Minds and Machines*, 18(2): 179–225, 2008.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

Matthew Schlegel, Adam White, Andrew Patterson, and Martha White. General value function networks. *arXiv preprint arXiv:1807.06763*, 2018.

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768, 2011.

Event segmentation reveals working memory forgetting rate

Anna Jafarpour *
University of Washington, Department of Physiology and Biophysics, Seattle, WA
annaja@uw.edu

Elizabeth A Buffalo
University of Washington, Department of Physiology and Biophysics, and National Primate Center, Seattle, WA
ebuffalo@uw.edu

Robert T Knight
University of California, Department of Psychology, Berkeley, CA and University of California, Department of Psychology, San Francisco, CA
rtknight@berkeley.edu

Anne GE Collins
University of California, Department of Psychology, and Helen Wills Neuroscience Institute, Berkeley, CA
annecollins@berkeley.edu

Abstract

We perceive the world as a sequence of events and fluidly segment it into episodes. Although people generally agree with segmentation, i.e., when salient events occur, the number of determined segments varies across the individuals. Converging evidence suggests that the working memory system plays a key role in tracking and segmenting a sequence of events (Zacks et al., 2007; Bailey et al., 2017). However, it is unclear what aspect of working memory is related to event segmentation and individual variability. Here, we tested whether the number of determined segments predicts working memory capacity, quantified as the number of items that can be kept in mind, or forgetting rate, which reflects how long each item is retained in the face of interference. Healthy adults (n=36, 18-27 years old) watched three movies with different storylines and performed a recognition memory test. They also participated in an image-action association learning task that was used to extract the individual's working memory capacity and forgetting rate (Collins et al., 2017). They then segmented the movies and performed a free recall task for the movies. Cross-task analyses showed that working memory forgetting rate is significantly related to event segmentation. Specifically, we found a U-shaped relationship between forgetting rate on the association learning task and the number of events segmented for the movies, suggesting that people with a higher forgetting rate may use distinct strategies for tracking events. Under-segmenters performed better in the temporal order recognition test for the movie with a linear and overarching storyline, while over-segmenters performed better on free recall. Working memory forgetting rate is a less studied parameter of the working memory system because of the high computational effort required to extract the parameter. These results support the possibility of using an individual's event segmentation performance to infer working memory forgetting rate.

Keywords: Perception, memory, reinforcement learning, human behavior

Acknowledgments: I thank Cassandra Lei, Crystal H. Shi, and Megan Schneider for data collection.

1 Experimental design

People watched three movies and then performed a temporal recognition test, where they saw two scenes of a movie and selected the order of the scenes. Movie 1 and 3 had an overarching storyline, but events in movie 2 were temporally interchangeable (like Tom&Jerry). Although temporal order recognition was better for movies with an overarching story than for movie 2, recognition performance was better than the chance level (50%) for all movies. People then performed an association learning task, learning stimulus-action association by trial and error (Collins et al. 2017; Figure 1). The actions were limited to three and the probability of each action being associated with a stimulus was equal. The number of stimulus-action associations was different in each block (ranging from 2 to 6). This task was used for estimating working memory capacity and forgetting rate. Then, people watched the movies again to subjectively segment them. They were instructed to press a key whenever a new event started and they knew that the aim was to segment the movie into episodes. Lastly, people wrote their memory of the movies (free recall).

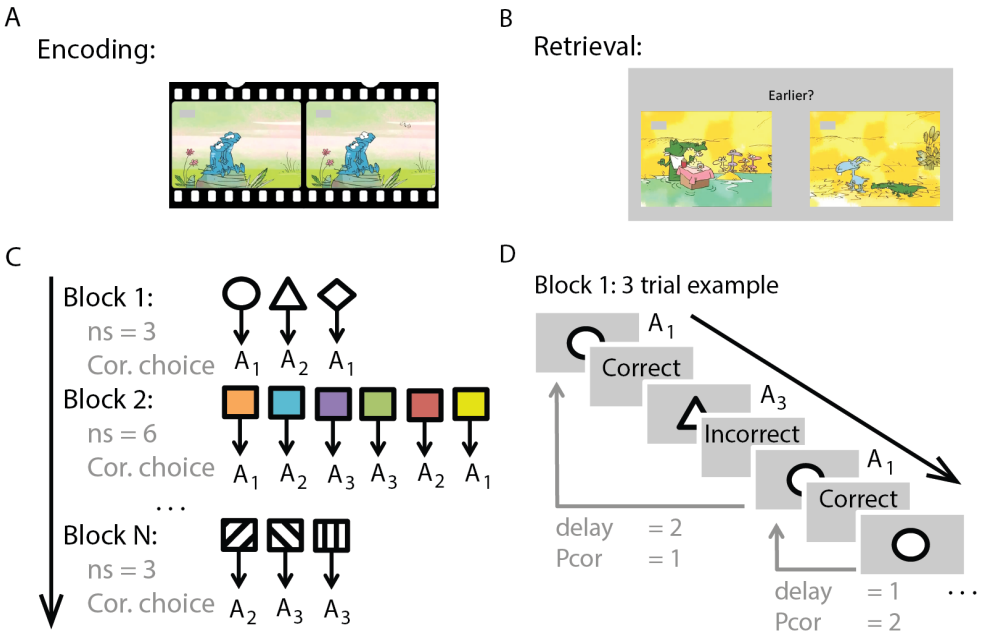


Figure 1 – Experimental design: The experiment consisted of four parts; the first two of which are depicted here. Temporal memory test: (A) At encoding, participants watched three mute movies. (B) At retrieval, participants were shown two frames of a movie and determined the temporal order of two movie frames by pressing the left or right key. There were 35 questions about the temporal order of events per movie. Association learning task: (C) Participants performed a block-design association learning task. In each block, participants learned by trial and error the association between a set of images and three possible actions (A1, A2, and A3); feedback was provided. The set size at each block was different (ranging from 2 to 6). (D) A 3-trial example of a learning block. Delay quantifies the number of intervening trials from the last time the stimulus was encountered, and Pcor quantifies the number of trials that the choice for the stimulus was correct.

2 Modeling

We applied two models. First, we used a logistic regressor to study the effect of the parameters - namely, set size, number of previously correct trials (Pcor), and delay (Figure 1D) - that are related to working memory, on trial by trial responses to the learning task and investigated the link between individuals' working memory limitation (the beta estimates of the logistic regressor) and the number of determined events. Then to clarify the aspect of working memory that is linked to individuals' event segmentation, we used a version of a reinforcement learning model that infers the working memory capacity and forgetting rate (RLWM, Collins et al. 2017). The validity of the RLWM model was tested by comparing the model fit to the fitting of RL models that do not have working memory components, and by comparing the simulated behavior of the RLWM to individuals' learning behavior.

We fit three RL models to the trial-by-trial responses for each subject. The simplest model was a two-parameters reinforcement learning (RL2) with delta rule learning (parameters were learning rate and the inverse temperature). The next model was had four-parameters (RL4); in addition to the two parameters of RL2, we estimated how much a person valued a correct response, irrespective of the reward by estimating the value for correct-but-not-rewarded items, i.e. r_0. The model also considered an undirected noise, $0 < \epsilon < 1$, in the stochastic action selection, to allow for choosing an action that did not have the highest value. The third model was a modified RLWM model (Collins et al., 2017).

RLWM model had 8 parameters and consisted of two components, working memory and RL. A working memory component with limited working memory *capacity, C*, and *forgetting rate,* $\emptyset_{WM}$. The Q value of the WM component was subject to decay with a forgetting rate, $0 < \phi < 1$, so for all the stimuli that are not current, $Q = Q + \phi(Q_0 - Q)$, where $Q_0 = \frac{1}{n_s}$. The RL component had a *learning rate, α,* value for an unrewarded correct response, r_0, undirected noise, ϵ, and a *forgetting rate,* $\emptyset_{RL}$ (β was set constant at 100). We also allowed for the potential lack of an impact of negative feedback ($\delta < 0$) by estimating a preservation parameter, $pers$. In that case, the learning rate is reduced by $\alpha = (1 - pers) \times \alpha$. Accordingly, $pers$ near 1 indicated lack of an impact of negative feedback (learning rate close to 0; high preservation of Q value), and $pers$ close to 0 indicated equal learning rate for positive and negative feedback.

The WM component was simulated as encoding of stimulus in a Q learning system, like the RL component but the outcome, r_t, was 1 for correct, 0 for incorrect (rather than the observed reward), the learning rate was set to 1 ($\alpha = 1$), and at most C stimuli could be remembered. We formulated the probability of a stimulus being in working memory as:

$$\text{If } r_t = 1, P_{WM}(r_t|s_t, a_t) = \min\left(1, \frac{C}{n_s}\right) \times Q_{wm}(s_t, a_t) + \left(1 - \min\left(1, \frac{C}{n_s}\right)\right) \times 1/n_a,$$

$$\text{if the } r_t = 0, P_{WM}(r_t|s_t, a_t) = \min\left(1, \frac{C}{n_s}\right) \times (1 - Q_{wm}(s_t, a_t)) + \left(1 - \min\left(1, \frac{C}{n_s}\right)\right) \times 1/n_a$$

, where n_a is the number of possible actions (= 3). In RL case,

$$\text{if the } r_t > 0, P_{RL}(r_t|s_t, a_t) = Q_{RL}(s_t, a_t)$$

$$\text{if the } r_t = 0, P_{RL}(r_t|s_t, a_t) = 1 - Q_{RL}(s_t, a_t).$$

A *mixture weight*, w_0, formulated how much each of the components were used for action selection. The weight was $w_0 \times \min\left(1, \frac{C}{n_s}\right)$ to represent the confidence in WM efficiency. This initialization reflects that participants are more likely to utilize WM when the stimulus set size is low. The overall policy was:

$P(a|s) = w_t(s) \times P_{WM}(a|s) + (1 - w_t(s)) \times P_{RL}(a|s)$.

A Bayesian model averaging scheme inferred the relative reliability of WM compared with the RL system over time, *t*:

$$w_{t+1}(s) = \frac{P_{WM}(r_t|s_t, a_t)\, w_t(s)}{P_{WM}(r_t|s_t, a_t)\, w_t(s) + P_{RL}(r_t|s_t, a_t)\,(1 - w_t(s))}$$

, where P_{WM} is the probability that action a is selected for stimulus s according to the WM component at time t and P_{RL} is the probability of action selection according to the RL component. We assumed that although the w_0 is the same for all stimuli, the development of mixture weight over time would be different for each stimulus, because the probability of retaining a stimulus in working memory or another retention system is not equal (Jafarpour et al., 2017). In all the RL models choosing an action utilized the expected reward value using the SoftMax choice rule.

The link between the number of determined events and working memory capacity and forgetting rate was tested by linear and quadratic model fitting. We used the Akaike Information Criterion (AIC) to select the best model considering the number of parameters used in each model.

3 Results and Discussion

Learning was near optimal for a low number of associations, but it decreased with increasing number of associations, reflecting working memory limitation (Figure 2A). The logistic regression confirmed this observation (Figure 2B) and the beta estimate for the interaction between Pcor and set size correlated with the number of determined events (movie 1: ranked r = 0.31, p = 0.064; movie 2: ranked r = 0.45, p = 0.006; movie 3: ranked r = 0.32, p = 0.057). To clarify the aspect of working memory that is linked to event segmentation, we used a RLWM model, which was the best fit among the RL models (pairwise t-test: RLWM and RL2: t(34) = 25.9, p < 0.001; RLWM and RL4: (t(34) = 9.63, p = 0.03). RLWM provided the individuals' working memory forgetting rate and capacity.

Event segmentation task showed that individual variability in number of determined events was consistent across the movies, irrespective to the storyline (Spearman ranked correlation: movie 1 vs. movie 2: r = 0.85, p < 0.001; movie 2 vs. movie 3: r = 0.86, p < 0.001; movie 1 vs. movie 3: r = 0.71, p < 0.001; Figure 2C for an example). We compared the estimated working memory capacity and forgetting rate to the number of determined events.

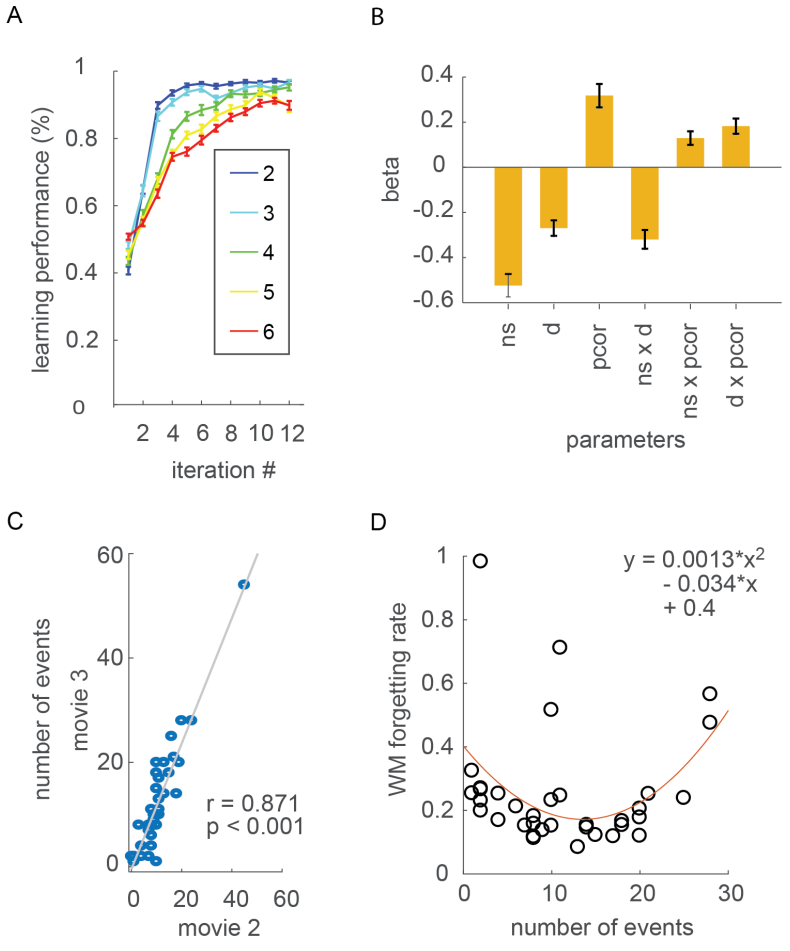


Figure 2 - (A) learning performance decreased with increasing learning set size. (B) logistic regression reveals the effect of working memory on learning performance. (C) inter-individual variability in the number of determined events was consistent across movies (correlation for movie 3 that had an overarching story and movie 2 that did not have an overarching story). The individual who determined a very high number of events was excluded from the future analysis, although the results are the same if we include her. (D) the cross-task result shows a U-shaped relationship between number determined events (movie 3) and working memory (WM) forgetting rate. Excluding the individual with a high estimation of forgetting rate improves the U-shaped fit.

The cross-task comparison showed that individuals who reported a very low or very high number of events had a higher working memory forgetting rate than others (Figure 2D). This effect was replicated across the three

movies. Excluding the participants with high estimated forgetting rate (two standard deviation more than the mean; outliers) improved the significance of the U-shaped relationship (movie 3: x: $p = 0.014$, x^2: $p = 0.012$).

The follow-up test showed that the over-segmenters (those who segmented one standard deviation more than the mean) recalled more about the movies than the under-segmenters (two-sample t-test $t(10) = -2.67$, $p = 0.023$); whereas, under-segmenters benefited from the overarching story of the movie and had a better temporal order memory than the under-segmenters (two sample t-test $t(12) = 3.26$, $p = 0.0068$). In a nutshell, by utilizing a cross-task design and RLWM model, we observed that the working memory forgetting rate reflects individual differences in event-segmentation. Here we used a separate task for estimating the working memory forgetting rate that is a less studied parameter of the working memory system due to the high computational effort. The data suggest that an individual's forgetting rate can be inferred from the segmentation rate.

Event segmentation is a domain-general cognitive control mechanism for chunking flow of events into episodes (Zacks et al., 2007) and extracting the structure of a flow of events (Jafarpour et al., 2019). The constitution of event segmentation is yet unclear. Here we observed that event segmentation reveals the working memory forgetting rate. A U-shaped relationship between the number of determined events on one task and the forgetting rate on another task showed that participants with a high forgetting rate used two different strategies for event perception. Critically, segmentation affects subsequent memory of events (Ezzyat and Davachi, 2014) and we showed that individual subsequent memory performance is variable depending on the segmentation rate. Under-segmenters remembered the order of schematic events better than over-segmenters, suggesting that they utilized schematic information to compensate for high working memory forgetting rate. In contrast, over-segmenters recollected more events than under-segmenters, reflecting the detailed chunking.

4 References

Bailey HR, Kurby CA, Sargent JQ, Zacks JM (2017) Attentional focus affects how events are segmented and updated in narrative reading. Mem Cogn 45:940–955.

Collins AGE, Albrecht MA, Waltz JA, Gold JM, Frank MJ (2017) Interactions Among Working Memory, Reinforcement Learning, and Effort in Value-Based Choice: A New Paradigm and Selective Deficits in Schizophrenia. Biological Psychiatry 82:431–439.

Ezzyat Y, Davachi L (2014) Similarity breeds proximity: Pattern similarity within and across contexts is related to later mnemonic judgments of temporal proximity. Neuron 81:1179–1189.

Jafarpour A, Griffin S, Lin JJ, Knight RT (2019) Medial orbitofrontal cortex, dorsolateral prefrontal cortex, and hippocampus differentially represent the event saliency. bioRxiv:285718.

Jafarpour A, Penny W, Barnes G, Knight RT, Duzel E (2017) Working-Memory Replay Prioritizes Weakly Attended Events. eNeuro:ENEURO.0171-17.2017.

Zacks JM, Speer NK, Swallow KM, Braver TS, Reynolds JR (2007) Event perception: a mind-brain perspective. Psychol Bull 133:273–293.

Safe Hierarchical Policy Optimization using Constrained Return Variance in Options

Arushi Jain
School of Computer Science
Mila - McGill University
Montreal, Canada
arushi.jain@mail.mcgill.ca

Doina Precup
School of Computer Science
McGill University, Google Deepmind
Montreal, Canada
dprecup@cs.mcgill.ca

Abstract

The standard setting in reinforcement learning (RL) to maximize the mean return does not assure a reliable and repeatable behavior of an agent in safety-critical applications like autonomous driving, robotics, and so forth. Often, penalization of the objective function with the variance in return is used to limit the unexpected behavior of the agent shown in the environment. While learning the end-to-end options have been accomplished, in this work, we introduce a novel Bellman style direct approach to estimate the variance in return in hierarchical policies using the option-critic architecture (Bacon et al., 2017). The penalization of the mean return with the variance enables learning safer trajectories, which avoids inconsistently behaving regions. Here, we present the derivation in the policy gradient style method with the new safe objective function which would provide the updates for the option parameters in an online fashion.

Keywords: option-critic,
safety,
constrained variance in return,
policy gradient

Acknowledgements

This research has received funding from Center of Research Institute in Montreal (CRIM).

1 Introduction

The objective function of maximizing the mean return does not offer any constraint on the distribution of the return, making it a vulnerable strategy for the risk-sensitive domains. The notion of avoiding risks arising from the stochastic nature of the environment (*inherent uncertainty*) using the constraint on the variance in return has been studied for a long time by the research community. Prashanth and Ghavamzadeh (2013); Sato et al. (2001); Tamar et al. (2012, 2016, 2013) constraint the indirect estimate of the variance using the second-order moment methods or directly estimated the cost-to-go returns with the updates provided after completing the entire trajectory. Sherstan et al. (2018) came up with a direct estimation of the variance in the λ-return using a Bellman operator in the policy evaluation methods. This work demonstrated the superiority of the direct estimator over the indirect approaches to estimate the variance.

Temporal abstraction provides a way to learn the policies in a hierarchical fashion which has been shown to improve exploration, robustness against model misspecification and increases the learning speed in transfer learning. Recently, option-critic architecture (Bacon et al., 2017) introduced an end-to-end style of learning the options. Jain et al. (2018) used the variance in the temporal difference (TD) error over the initial state-option pair distribution to estimate the controllable states in the option-critic.

In this work, we came up with a novel hierarchical safe policy learning approach in the option-critic architecture where the hierarchical policies are learned by penalizing the direct estimate of the variance in return extending from Sherstan et al. (2018) in a control setting. We seek to maximize the mean return and minimize the direct estimate of the variance in return given an initial state-option pair distribution in the policy gradient style.

2 Background

In a Markov Decision Process (MDP), an agent takes an action $a \in A$, transitions from state S_t to state S_{t+1}, and receives an immediate reward R_{t+1} from the environment. The expected reward is $r(S_t, A_t) = \sum_{r \in \mathbb{R}} r \sum_{s'} P(s', r | S_t, A_t)$ where $r : S \times A \to \mathbb{R}$. The environment dynamics is modeled by $P(S_{t+1}|S_t, A_t)$, where $P : S \times A \times S \to [0, 1]$. A stochastic policy $\pi(A_t|S_t)$ determines the probability of taking an action in a given state. The MDP is represented by a tuple $\langle S, A, P, r, \gamma \rangle$, where $\gamma \in [0, 1]$ is a factor discounting the future rewards.

2.1 Option-Critic

The option-critic architecture (Bacon et al., 2017), an option $w \in W$ is defined as a tuple of $\langle I_w, \pi_w, \beta_w \rangle$; where I_w contains the initial set of states where an option can start, π_w is the option policy defining a distribution over action space and β_w determines the termination probability of an option in a state. The policy over the options is denoted by $\mu(w|s)$ describing the distribution over options given a state. Let $\Theta = [\theta, \nu, \kappa]$ be the parameters of intra-option policy π_w, termination condition β_w and policy over options μ respectively. $J_{\pi,\mu}$ denotes the objective function of maximizing the mean return. The intra-option policy gradient (Bacon et al., 2017) update is:

$$\nabla_\theta J_{\pi,\mu}(\Theta) = \mathbb{E}_{\pi,\mu}[\nabla_\theta \log \pi_\theta(A_t|S_t, W_t) Q_{\pi,\mu}(S_t, W_t, A_t)],$$

and the termination gradient (Bacon et al., 2017) is given by:

$$\nabla_\nu J_{\pi,\mu}(\Theta) = \mathbb{E}_{\pi,\mu}[-\nabla_\nu \beta_\nu(S_{t+1}, W_t) A_{\Theta,Q}(S_{t+1}, W_t)]$$

where, $A_{\Theta,Q}(S_{t+1}, W_t) = Q(S_{t+1}, W_t) - V(S_{t+1})$ is the advantage function describing the importance of an option value over the mean value. In the following work we assume that all the options can be started from any state ($I_w \in S \,\forall\, w \in W$).

3 Safety in Option-Critic

Taking inspiration from the notion of safety in the actor-critic framework using the constraint variance in return (Jain et al., 2019), we similarly derive the safe framework in the option-critic. Our notion of *safety* emphasizes minimizing the *erratic* or the *harmful* behavior of an agent in the environment (Amodei et al., 2016). The higher is the variance in return from a state; the higher would be the uncertainty in the value estimate of that state. Uncertainty in the value estimate of a state reflects an inconsistent behavior of the agent in that particular state. Considering that the irregular or sudden behavior is classified as unsafe, potentially, the unsafe states would exhibit higher variance in return.

Let the return be denoted by

$$G_{t,\pi,\mu} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = R_{t+1} + \gamma G_{t+1,\pi,\mu}.$$

We consider $Z_t = (S_t, W_t)$ as an augmented state space - a space of state-option pair. Here, the transition matrix over the augmented state space is given by:

$$P(z'|z, a) = P(s'|s, a)[(1 - \beta_\nu(s', w)) \mathbb{1}_{w=w'} + \beta_\nu(s', w) \mu_\kappa(w'|s')] \tag{1}$$

The rewards are coming from a base MDP, where we write $r(z,a,z') = r(s,a)$. Since, $\sum_{z'} P(z'|z,a) = 1$, therefore, the reward model is defined as:

$$r(s,a) = \mathbb{E}_{\pi,\mu}[R_{t+1}|S_t = s, A_t = a] = \sum_{z'} P(z'|z,a) r(z,a,z')$$

Lemma 1. $\mathbb{E}_b[\gamma\lambda\delta_{t,\pi}\big(\rho_{t+1}G^{\lambda}_{t+1,\pi} - \mathbb{E}_b[\rho_{t+1}Q_\pi(S_{t+1},A_{t+1})|S_t = s, A_t = a]\big)\big|S_t = s, A_t = a] = 0.$

Proof. The proof of the lemma is given in Jain et al. (2019). Here λ is trace decay parameter and $\delta_{t,\pi}$ is the 1-step TD error. □

Theorem 1. *The Bellman equation for the variance in the return from a given augmented state-action pair is:*

$$\sigma_{\pi,\mu}(z,a) = \mathbb{E}_{\pi,\mu}\big[\delta^2_{t,\pi} + \bar{\gamma}\sigma_{\pi,\mu}(Z_{t+1},A_{t+1})\big|Z_t = z, A_t = a\big] \tag{2}$$

where $\bar{\gamma} = \gamma^2$ and δ_t is the 1-step TD error.

Proof. On expanding $G_{t,\pi,\mu} - Q_{\pi,\mu}(z,a)$,

$$\begin{aligned}
G_{t,\pi,\mu} - Q_{\pi,\mu}(z,a) =& R_{t+1} + \gamma G_{t+1,\pi,\mu} - Q_{\pi,\mu}(z,a) \\
=& R_{t+1} + \gamma \sum_{z',a'} P(z'|Z_t,A_t)\pi_\theta(a'|z')Q_{\pi,\mu}(z',a') - Q_{\pi,\mu}(z,a) \\
&+ \gamma\{G_{t+1,\pi,\mu} - \sum_{z',a'} P(z'|Z_t,A_t)\pi_\theta(a'|z')Q_{\pi,\mu}(z',a')\} \\
=& \delta_t + \gamma\Big(G_{t+1,\pi,\mu} - \mathbb{E}_{\pi,\mu}[Q_{\pi,\mu}(Z_{t+1},A_{t+1})|Z_t = z, A_t = a]\Big)
\end{aligned} \tag{3}$$

Similar to Jain et al. (2019), let the variance for the augmented state-action pair be given as:

$$\begin{aligned}
\sigma_{\pi,\mu}(z,a) =& \mathbb{E}_{\pi,\mu}\left[(G_{t,\pi,\mu} - \mathbb{E}_{\pi,\mu}[G_{t,\pi,\mu}|Z_t = z, A_t = a])^2|Z_t = z, A_t = a\right] \\
=& \mathbb{E}_{\pi,\mu}\left[(G_{t,\pi,\mu} - Q_{\pi,\mu}(z,a))^2|Z_t = z, A_t = a\right] \\
=& \mathbb{E}_{\pi,\mu}\left[\Big(\delta_t + \gamma(G_{t+1,\pi,\mu} - \mathbb{E}_{\pi,\mu}[Q_{\pi,\mu}(Z_{t+1},A_{t+1})|Z_t = z, A_t = a])\Big)^2|Z_t = z, A_t = a\right] \\
=& \mathbb{E}_{\pi,\mu}\left[\delta^2_t|Z_t = z, A_t = a\right] + \gamma^2\,\mathbb{E}_{\pi,\mu}\left[(G_{t+1,\pi,\mu} - \mathbb{E}_{\pi,\mu}[Q_{\pi,\mu}(Z_{t+1},A_{t+1})|Z_t = z, A_t = a])^2|Z_t = z, A_t = a\right] \\
&+ 2\gamma\,\mathbb{E}_{\pi,\mu}\left[\delta_t(G_{t+1,\pi,\mu} - \mathbb{E}_{\pi,\mu}[Q_{\pi,\mu}(Z_{t+1},A_{t+1})|Z_t = z, A_t = a])\right]
\end{aligned} \tag{4}$$

Using the Lemma 1, by substituting $\rho, \lambda = 1$ and changing the state S as an augmented state Z, the third term in the above (4) goes to 0. This leads the variance to $\sigma_{\pi,\mu}(z,a) = \mathbb{E}_{\pi,\mu}\big[\delta^2_{t,\pi} + \bar{\gamma}\sigma_{\pi,\mu}(Z_{t+1},A_{t+1})\big|Z_t = z, A_t = a\big]$. □

The new safe objective function is defined as:

$$\boldsymbol{J(\Theta)} = \mathbb{E}_{z\sim d}\big[Q_\Theta(z) - \psi_z\,\sigma_\Theta(z)\big],$$

where d describes the initial state-option distribution and ψ_z is the regularizer for the variance penalty term which is a function of the augmented state space.

Theorem 2. *(Safe Intra-Option Policy Gradient Theorem) Given Markov options, $\pi_{w,\theta}$ policy differentiable in parameter θ, the gradient of the objective function J w.r.t. θ starting from state s and option w is:*

$$\nabla_\theta J(\Theta) = \mathbb{E}_{d,\Theta}[\sum_a \nabla_\theta \pi_\theta(a|Z_t)\big(Q_{\pi,\mu}(Z_t,a) - \psi_{Z_t}\,\sigma_{\pi,\mu}(Z_t,a)\big)]$$

Proof. The gradient of the $\sigma_\Theta(z)$ w.r.t. θ is calculated in a similar fashion as the gradient of $Q_\Theta(z)$ w.r.t. θ in the *Intra-option Policy Gradient Theorem* (Bacon et al., 2017). □

The gradient update for the intra-option policy moves in the direction to maximize the mean return value and minimize the variance in the return.

Theorem 3. *(Safe Termination Gradient Theorem) Given Markov options, $\beta_{w,\nu}$ termination function differentiable in parameter ν, the gradient of the objective function J w.r.t. ν starting from state s and option w is:*

$$\nabla_\nu J(\Theta) = \mathbb{E}_{d,\Theta}[-\nabla_\nu \beta_\nu(S_{t+1},W_t)\big(A_{\pi,\mu,Q}(S_{t+1},W_t) - \psi_z\,A_{\pi,\mu,\sigma}(S_{t+1},W_t)\big)]$$

where $A_{\pi,\mu,\sigma}(S_{t+1},W_t) = \sigma_\Theta(S_{t+1},W_t) - \sigma_\Theta(S_{t+1})$ is the advantage function for the variance similar to the value function.

Proof. The gradient of the $\sigma_\Theta(z)$ w.r.t. ν is calculated in a similar fashion as the gradient of $Q_\Theta(z)$ w.r.t. ν in the *Termination Gradient Theorem* (Bacon et al., 2017). □

Similar to the Option-Critic, when the advantage of the value function is positive for an option, the gradient for the termination descends. On the other hand, the positive advantage function for the variance makes the gradient update for the termination ascent. It matches with the intuition, when the variance of an option is higher than the average variance, it would be desirable to terminate the option and choose a better option with a lower variance.

Theorem 4. *(Safe Policy over Options Gradient Theorem) Given Markov options, μ_κ policy over options differentiable in parameter κ, the gradient of the objective function J w.r.t. κ starting from state s and option w is:*

$$\nabla_\kappa J(\Theta) = \mathbb{E}_{d,\Theta}[\beta_\nu(S_{t+1}, W_t) \sum_{w'} \nabla_\kappa \mu_\kappa(w'|S_{t+1}) \big(Q_\Theta(z') - \psi_{z'}\, \sigma_\Theta(z')\big)]$$

Proof. Let 1-step augmented state transition using (1) be: $P^{(1)}_{\bar\gamma}(Z_{t+1}|Z_t) \overset{\text{def}}{=} \bar\gamma \sum_a \pi_\theta(a|Z_t) P(Z_{t+1}|Z_t, a)$. Similarly, the k-step transition would be defined as: $P^{(k)}_{\bar\gamma}(Z_{t+k}|Z_t) \overset{\text{def}}{=} P^{(1)}_{\bar\gamma}(Z_{t+k}|Z_{t+k-1}) \times P^{(k-1)}_{\bar\gamma}(Z_{t+k-1}|Z_t)$. The gradient of the variance w.r.t. κ parameter following (2),

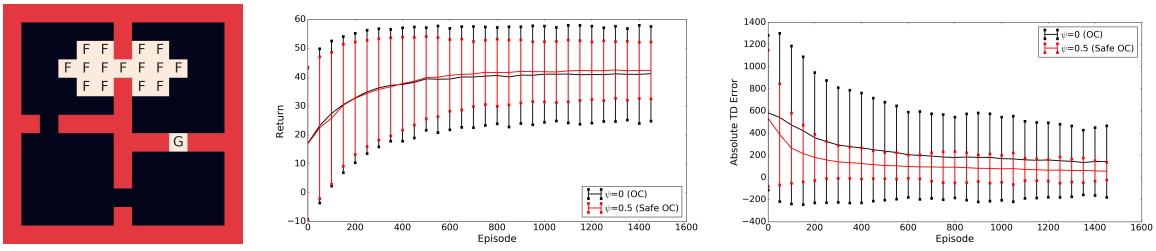
$$\begin{aligned}
\nabla_\kappa \sigma_\Theta(z) =& \nabla_\kappa \Big[\sum_a \pi_\theta(a|z) \bar\gamma \sum_{s'} P(s'|s,a) \big[(1-\beta_\nu(s',w))\sigma_\Theta(s',w) + \beta_\nu(s',w) \sum_{w'} \mu_\kappa(w'|s')\sigma_\Theta(s',w')\big]\Big] \\
=& \sum_a \pi_\theta(a|z) \bar\gamma \sum_{s',w'} P(s'|s,a) \big[(1-\beta_\nu(s',w)) \mathbb{1}_{w=w'} + \beta_\nu(s',w)\mu_\kappa(w'|s')\big] \nabla_\kappa \sigma_\Theta(s',w') \\
&+ \sum_a \pi_\theta(a|z) \bar\gamma \sum_{s',w'} P(s'|s,a)\beta_\nu(s',w) \sum_{w'} \nabla_\kappa \mu_\kappa(w'|s') \sigma_\Theta(s',w') \\
=& \sum_{k=0}^{\infty} \sum_{z'} P^{(k)}_{\bar\gamma}(z'|z) \sum_{a'} \pi_\theta(a'|z') \sum_{s''} \bar\gamma P(s''|s',a') \beta_\nu(s'',w') \sum_{w''} \nabla_\kappa \mu_\kappa(w''|s'') \sigma_\Theta(s'',w'')
\end{aligned}$$

Similarly, the gradient of the $Q_\Theta(z)$ value function can be derived similarly, leading to the proof. □

The above theorem states that the gradient of the policy over the options is updated in the direction of maximizing the expected Q-value and minimizing the variance function achieved from all other possible options after termination of the current option.

4 Experiment

Grid-World: We experiment in the classic grid-world four rooms (FR) environment (Bacon et al., 2017). To test safety, we created a variable reward frozen patch (F) in one of the hallway generated from $\mathcal{N}(0, 8)$ distribution. The rest of the states are given a 0 reward. Agent receives a reward of 50 on reaching the goal (G) (See Fig. 1a). γ is kept as 0.99. The step size of value function, variance function, intra-option policy, termination, policy over options are $1.0, 2e-3, 1e-3, 5e-3, 1e-4$ respectively for both option-critic ($\psi_z = 0$) and safe option-critic ($\psi_z = 0.5$) $\forall z \in Z$. Fig. 1b and Fig. 1c depict the performance in the FR environment.

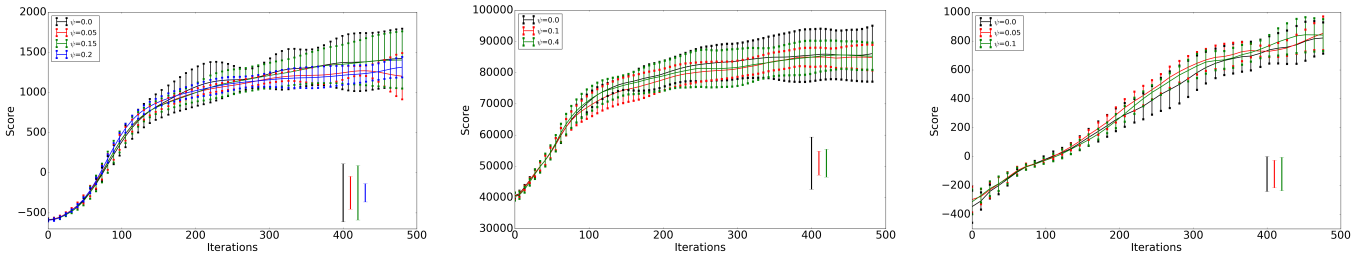


(a) Environment (b) Learning curve (c) Absolute TD error

Figure 1: **Performance in the FR**: Shows the performance averaged over 50 trials where the vertical bands depict the std. dev.. Shows b) the return, and c) sum of the absolute TD error. The safe policy (red) has a smaller standard deviation as compared to the baseline (black) signifying safety helps an agent to avoid variance inducing regions.

Continuous State-Action Space: Here we performed the experiments in Mujoco environments to test the real-world use case of introducing safe trajectories while learning in an environment. We implemented our safe algorithm over existing

proximal policy option-critic (PPOC) (Klissarov et al., 2017). We compare the performance of the agent using both the baseline PPOC and Safe-PPOC in Fig. 2. The videos[1] compare the performance of the agent using both the algorithms in the Mujoco environments.



(a) Half Cheetah (b) Humanoid Standup (c) Ant

Figure 2: **Performance in Mujoco**: Learning curve average over 10 runs where vertical bands depict the std. dev.. The vertical bars at right most corner display the std. dev. in performance over the last 50 iterations. The variance regularized PPOC ($\psi > 0$) helps in reducing the variation across multiple seed values leading to a more consistent performance.

5 Conclusion & Future Work

This work aims to introduce the constraint over the variance in return to the existing option-critic architecture in order to incorporate responsible behavior in the risk-sensitive domains. Firstly, we propose a direct estimator of the variance in the hierarchical policy framework. Then, we establish a method to learn a safe and reliable policy in option-critic, which uses the above direct estimator of the variance to avoid unpredictably behaving regions. The above framework is generic, which makes no assumption about the environment, making it a simple strategy to combine with the current policy gradient techniques. The *future work* is to experiment with more different environments like Atari to understand the scalability of the safe algorithm.

References

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *AAAI*, pages 1726–1734.

Jain, A., Jain, A., Khetarpal, K., , Aboutalebi, H., and Precup, D. (2019). On-policy and off-policy actor-critic with constrained return variance. In *Under Submission*.

Jain, A., Khetarpal, K., and Precup, D. (2018). Safe option-critic: Learning safety in the option-critic architecture. *arXiv preprint arXiv:1807.08060*.

Klissarov, M., Bacon, P.-L., Harb, J., and Precup, D. (2017). Learnings options end-to-end for continuous action tasks. *arXiv preprint arXiv:1712.00004*.

Prashanth, L. and Ghavamzadeh, M. (2013). Actor-critic algorithms for risk-sensitive MDPs. In *Advances in neural information processing systems*, pages 252–260.

Sato, M., Kimura, H., and Kobayashi, S. (2001). TD algorithm for the variance of return and mean-variance reinforcement learning. *Transactions of the Japanese Society for Artificial Intelligence*, 16(3):353–362.

Sherstan, C., Ashley, D. R., Bennett, B., Young, K., White, A., White, M., and Sutton, R. S. (2018). Comparing direct and indirect temporal-difference methods for estimating the variance of the return. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 63–72.

Tamar, A., Di Castro, D., and Mannor, S. (2012). Policy gradients with variance related risk criteria. In *Proceedings of the twenty-ninth international conference on machine learning*, pages 387–396.

Tamar, A., Di Castro, D., and Mannor, S. (2016). Learning the variance of the reward-to-go. *Journal of Machine Learning Research*, 17(13):1–36.

Tamar, A., Xu, H., and Mannor, S. (2013). Scaling up robust MDPs by reinforcement learning. *arXiv preprint arXiv:1306.6189*.

[1]The performance videos of the agent in PPOC and Safe PPOC in Mujoco domains is here.

Making Meaning: Semiotics Within Predictive Knowledge Architectures

Alex Kearney
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
kearney@ualberta.ca

Oliver Oxton
Department of Philosophy
University of Waterloo
Waterloo, Ontario, Canada
opo.xton@gmail.com *

Abstract

Within Reinforcement Learning, there is a fledgling approach to conceptualizing the environment in terms of predictions. Central to this predictive approach is the assertion that it is possible to construct ontologies in terms of predictions about sensation, behaviour, and time—to categorize the world into entities which express all aspects of the world using only predictions. This construction of ontologies is integral to predictive approaches to machine knowledge where objects are described exclusively in terms of how they are perceived. In this paper, we ground the Pericean model of semiotics in terms of Reinforcement Learning Methods, describing Peirce's Three Categories in the notation of General Value Functions. Using the Peircean model of semiotics, we demonstrate that predictions alone are insufficient to construct an ontology; however, we identify predictions as being integral to the meaning-making process. Moreover, we discuss how predictive knowledge provides a particularly stable foundation for semiosis—the process of making meaning—and suggest a possible avenue of research to design algorithmic methods which construct semantics and meaning using predictions.

Keywords: Reinforcement Learning, General Value Functions, Philosophy, Epistemology, Semantics.

Acknowledgements

Thanks to Patrick Pilarski, Johannes Günther, Melissa Woghiren, Anna Koop, and Niko Yasui for feedback on early drafts of this manuscript. Thanks to Dylan Jones for insightful discussion. This work was supported in part by the Alberta Machine Intelligence Institute, Alberta Innovates, the Natural Sciences and Engineering Research Council, the Canada Research Chairs program, and by Borealis AI through their Global Fellowship Award.

*For additional discussion on the epistemology of predictive knowledge and how we can view predictive knowledge architectures as having knowledge, please refer to Kearney & Pilarski, "When is a Prediction Knowledge?", accepted to RLDM 2019.

1 Predictive Approaches to Machine Knowledge

A foundational problem of machine intelligence is being able to conceptualize the environment—to construct ontologies of categories and concepts which enable meaningful decision making and problem solving. Furthermore, it is normally assumed that such ontologies are dependent upon knowledge; and so it is no surprise that each approach to machine intelligence—from expert systems to machine learning methods—aims to build systems approaching human-level understanding of their environment based on differing assumptions about what knowledge is and what counts as it. Within Reinforcement learning, there is a fledgling approach to knowledge building sometimes called *predictive knowledge*: a collection of learning methods and architectural proposals which seek to describe the world exclusively in terms of sensation, behaviour, and time (Sutton, 2009).

What separates predictive knowledge from other machine intelligence proposals is a focus on, and requirement of, methods which have three capacities: the ability to 1) self-verify knowledge through continual interaction with the environment, 2) describe knowledge exclusively in terms of observations from the environment, and 3) scale learning methods (Sutton et al., 2011). While each of the requirements is important to operationalizing predictive knowledge, they are in service of a greater, unmentioned requirement: the ability to construct an ontology through interaction with the environment. Many other learning methods assume that ontological categories are given to the agent, either by hand-crafting properties and relationships to describe the world (as is done in knowledge bases), or providing explicit ontological categorization during supervised learning. Predictive Knowledge is liberated from this engineering process by focusing on learning methods which construct their own categories, properties, and relationships to describe the environment.

While promising, it is unclear to what extent ontologies constructed through predictive knowledge frameworks accomplish the task of conceptualizing the environment. Certainly there is some degree of success in using predictions to support control and decision-making (Modayil and Sutton, 2014; Edwards et al., 2016; Günther et al., 2016), but there are also roadblocks to progress which seem to indicate a problem with the foundations of the framework. For instance, tasks such as reasoning about objects in terms of sensorimotor experience in predictive terms (Koop, 2008) has proven to be exceptionally challenging—a task which is relatively simple for supervised learning systems, and other learning methods with hand-crafted ontologies. This difficulty to express general concepts using predictions is sometimes referred to as the *abstraction gap*.

At root, and so in service of the three requirements above, the predictive knowledge project proposes that value functions hold the meaning necessary to construct a successful ontology.[1] This paper will proceed by proposing a framework for understanding meaning which will present a challenge to this stated understanding of value functions, the technical implications of which will then be discussed. Importantly, we find that Predictive Knowledge does not yet meet the threshold for meaning that is necessary to accomplish the ontological construction sought after by the project of Predictive Knowledge.

2 The Building Blocks of Meaning: a Less Deadly Triad

There is a long standing approach to the construction of meaning in academic literature outside of Computer Science and Machine Intelligence; namely, semiotics. Despite particular theoretical differences, the basic idea behind positions in semiotics is that meaning is a relational product of 'signifiers' (or, words and language) and the 'signified' (or, objects and the world). For instance, Saussure argued that the meaning for some word was defined by its relative place in the broader structure of the relevant language—i.e. *dog* does not mean *cat* (and vice-versa) precisely because the structure of the English language has unique places, laid out by interpreters through practice, wherein *dog* and *cat* cannot be interchanged. Thus, there is the signified (one's actual pet) and it stands in relation to a structured ontology of signifiers (of words one might use, or not use, to describe their pet).

Of course, put so broadly, semiotics as an approach to linguistic meaning is simultaneously compatible with a wide variety of other methodological approaches and philosophical commitments, and prone to questioning and conflict. Consequently, to see how semiotics may help design machine learning methods capable of independently constructing ontologies, the discussion must be narrowed from the broader theoretical terrain

[1] This is made clear in some conceptual projects(Sutton, 2009; Koop, 2008; Sutton et al., 2011); however, it is worth noting that in engineering projects which focus on using predictions to support decision making, it is likely that the claim of predictions as being relevant is likely only implicit, if intended at all.

to one particular view. To that end, we focus on the Peircean model of semiotics for two reasons: 1) Peircean semiotics is not limited to the domain of language, but instead, tackles the broadest domain possible with the notion of a *sign* (CP 1.339)[2], making it applicable to the analysis of machine intelligence methods; and 2) Peircean semiotics particularly emphasizes the process of agent interaction rather than the resulting 'language' or ontological structure (CP 1.341), making it uniquely suitable to analysis of Reinforcement Learning methods.

At a macroscopic level, Peirce's semiotics is often described as a triadic relation between an object, a signifier, and an interpretation. For example, a bonfire at a campsite could be an object; the smoke it gives off could be its signifier; and *the conclusion that one draws* (or, interpretation) could be that "people are in the forest." One may note that under the Peircean model, signifiers are not only words or the perspective of agents, but can also be objects in the world. This leads to two immediate confusions: 1) that there is a multiplicity to the interpretation of signs (it is not clear the the 'right' signifier is the smoke, or the fire, or that the 'right' interpretation is that there are people in the forest); and 2) that signs are not isolated or fixed, but instead are linked together (a 'complete' sign between smoke, fire and the interpretation of people, might itself be the signifier of another larger, more complex sign). However, despite these points of potential confusion, the model nonetheless provides a framework for evaluating meaning: smoke *means* people are present due to the combination of relations inherent between smoke and fire, fire and people, *and* the interpretive step an agent takes in relating their environment and experiences to these relations. The crux of this model is thus the third aspect of agent interpretation rather than simply the sets of relations between phenomena.

Indeed, to go into further detail, one will find that the triadic model is derivative of Peirce's Theory of Mind.[3] What can be shown is that the semiotic model depends upon, what is normally called, the *Three Categories*; and which we will refer to here as *Sensation*, *Perception*, and *Generality*.[4] First, before abstractions—or conceptualizations—can be constructed, there must be the information or sensation from which one can construct the abstraction, absent of any categorization, modelling, or understanding. Thus, *Sensation* is the observation an agent receives from the environment without further analysis, comparison, or relation. In the linguistic semiotic model, this would be the smoke *as smoke*—a signifier without a corresponding object or interpretation, something merely sensed.

Of course, our concepts and thoughts are not simply composed of raw, unprocessed sensation. A sensation of smoke and a sensation of fire are related through our perception and environment construction *regardless of any particular meaning.* Consider, for example, how classical conditioning describes fixed responses to stimuli, such as blinking, which do not require higher level conceptual cognition. Thus the second component of the triad, *Perception*, describes both the properties which our environment is in terms of and how each property relates to different sensations. Within predictive knowledge and machine learning, this notion of *Perception*, can be found in many places: i.e. a prediction's estimate, or the value of a state action pair.

Now, it may seem unclear why one needs a third part. There are moments sensed and perceptions which relate them; what else could there be, or must there be? Some reflection, however, may reveal the shortcomings of this diadic relationship. To think again of our campsite, one's sensation of smoke may bring about the perceptual relation of fire—the two go together after all, like blinking—but there are many further experiences which come with fire. Sometimes there really is a campsite and people roasting marshmallows, other times lightning strikes, or dry heat and unkempt brush may bring an unwelcome end to celebratory fireworks. Which conclusion one draws from the sensation of smoke—that is, what the smoke *means*—is thus the application of a broader general concept (say, *campfire* as compared to *forest fire*). The useful selection of one broader concept over another may require a rich background of experience and learned relations (between, say, the volume of smoke and the colour of the horizon, or the smell of burning pine), but is nonetheless an active cognitive step wherein a *general notion* is applied to a particular case—that the source of *this* smoke, and so this fire, is people and not lightning. This third category, the relation of general patterns to particular instances, is what we have called *Generality*.

[2]We refer to *Collected Papers of Charles S. Peirce* by CP m.n, where m is the volume number and n is the paragraph number, as is custom in Peircean scholarship.

[3]The full scope of which is, unfortunately, beyond this paper.

[4]Peirce refers to these categories as Firstness, Secondness, and Thirdness; however, despite the systematic function that these categories play throughout his theories, we eschew this naming schema for both clarity and applicability. See CP 1.300 onwards for relevant discussion.

3 Is Learning a GVF a Semiotic Process?

Thus far we have presented the parallels between AI and semiotics speculatively and hesitantly, but we will now develop an extended example covering predictive knowledge and the Three Categories. To do so, we take a General Value Function (GVF) (White, 2015)—the most basic mechanism of many predictive knowledge proposals—and evaluate whether learning an approximate value function can be seen as a triadic relationship; we evaluate whether predicting is a process which produces meaning.

GVFs make predictions estimating the *value*, or expected discounted sum of a signal C defined as $G_t = \sum_{k=0}^{\infty}(\prod_{j=1}^{k}(\gamma_{t+j}))C_{t+k+1}$. Value for some state ϕ is estimated with respect to a specific policy π, discount function $0 \leq \gamma \leq 1$, and cumulant c, such that $v(\phi;\pi,\gamma,c) = \mathbb{E}_\pi[G_t|S_t = \phi]$. Using GVFs, we can ask questions such as "How long will it take me to bump into a wall if I keep walking forwards"? These GVFs are typically learnt online through interaction between an agent and its world over discrete time-steps. On each time-step $t = 0, 1, 2, ..., n$ the agent receives a vector o_t that describes what is sensed, and takes an action a_t. Prior to use or feature construction, the observations o_t received by the system are *Sensation*: the first component of the Three Categories.

The observations, with some function approximator, are used to produce the *agent state*: a feature vector $\phi_t : o_t \to \mathbb{R}^n$ which describes the environment from the agent's perspective[5]. This state ϕ_t is used in conjunction with some learning method to estimate the discounted sum G_t of future signals C. For our example, we consider Temporal-Difference (TD) learning (Sutton, 1988); however, our conclusions will generalize to other policy evaluation methods. When performing TD learning, we maintain some weight parameters $w \in \mathbb{R}^n$ which when combined with the current state produce the estimated return $v_\pi(\phi_t) = w_t^\top \phi_t$. On each step, at each instant, the weights are changed proportional to the TD error $\delta_t = C_{t+1} + \gamma_{t+1} v_\pi(\phi_{t+1}) - v_\pi(\phi_t)$. When the weights are updated by $w_{t+1} = w_t + \alpha_t \delta_t \phi_t$, the relation between o_t, a_t, and c_t is updated based on some response from the environment. For any given state ϕ_t, the estimate $v_\pi(\phi_t)$ forms a relation between what is sensed o_t, the actions taken a_t and the signal being predicted c_t; thus, value estimates form the second component of the Three Categories: *Perception*.

Having come to the end of the process of specifying and learning a GVF, one may wonder where *Generality* exists in predictive knowledge. After all, many claim that a single prediction has meaning[6], but we have so far only identified *Sensation* and *Perception*. While a GVF may capture a prediction for any given state, *generalizing* over observations through some function approximation, it does not capture *Generality*. When a prediction is formed as a GVF, our expectation of future signals slurs over all experience, making it impossible to relate manifestations of instances of signals in order to compare and contrast them. Using GVFs alone, we are incapable of, say, identifying that a wall bumped into is the very same as the one we bumped into both 10 time-steps ago and 100 time-steps ago: we may only say how close an observation was to the expectation of the whole of an agent's experience in that particular agent-state.

This limitation in expressing the Three Categories invites us to wonder whether the notion of Sensation, Perception, and Generality is a productive one: does framing predictive knowledge as a semiotic process help us better understand machine intelligence? As we previously introduced, there are other approaches to semiosis, some of which do not depend on *Generality*[7]. Simply finding our methods to be meaningful does not obliviate the limitations of existing predictive knowledge methods. Predictive knowledge frameworks can be construed as constructing meaning under other definitions of semiotics; however, declaring our methods to be sufficient does not help predictive knowledge systems cross the abstraction gap and express concepts which are at present elusive—a declaration of meaning would not suddenly enable Predictive Knowledge methods to reason about generality, or make it any clearer how notions such as objects would be formalized in a predictive setting.

How, then, do we surmount the gap between abstract generalities and the relations which inform them? While predictions alone are insufficient, it may be possible to express generalities by constructing models using predictions A model-based method which explicitly defines state-action transitions relates not only one state s_t to another s_{t+1}, but relates states in terms of all the possible transitions given all the possible actions

[5]It is worth noting that input observations o_t could include not just immediate sensor feedback, but also the previous action, historical information, or internal signals generated from learning.

[6]See Sutton et al. (2011) for an example argument for GVFs as having explicit semantics, and both Koop (2008) and White (2015) for additional discussion of predictions as inherently meaningful.

[7]For example, (Barbieri, 2007) presents a variety of semiotic models in application to cell biology.

which could have been taken in s_t. A model which is able to interrelate many predictions such that contexts can be compared and contrasted could be considered semiotic. For these reasons, difficulty in constructing generalities does not belong to insufficient learning methods, or poor state construction (although they do impact progress). The difficulty of constructing abstractions results from an inability to interrelate what is learned: it is a problem of how we structure predictive knowledge architectures, not how we learn them.

Does learning a prediction encompass the entirety of a semiotic process? No; however, predictive knowledge could play a central role in a process which is semiotic. GVFs provide a robust and flexible foundation for semiosis by playing the part of *Perception*. By focusing on incremental learning methods which are specified in terms of behaviour, GVFs describe a class of predictions which are uniquely suited to be Perception: methods which are not ontologically constrained by labels—or, what predictive knowledge agents can conceptualize constrained by the reality of the environment they inhabit, not the labels provided for training. Moreover, GVFs have proven themselves to be practically useful for reactive behaviour such as prosthetic control (Edwards et al., 2016), lazer welding (Günther et al., 2016), and robotics (Modayil and Sutton, 2014)—crucial steps in demonstrating that GVFs are useful in informing decision-making about the environment, although insufficient for constructing an ontology of the world.

4 Conclusion: Taking Stock of the Predictive Knowledge Project

Predictive knowledge describes a collection of proposals for constructing machine knowledge which assert that all world knowledge can be described as predictions about sensation, behaviour, and time. In this paper, we take a first look at the construction of semantics and meaning in predictive knowledge by evaluating whether or not learning a General Value Function can be construed as a semiotic process. We demonstrate that GVFs can be seen to fulfill the first two components of semiosis: *Sensation* and *Peception*; however, predictions fall short of providing the third component, *Generality*. As a result, predictions do not have meaning independent of any other process. While predictive knowledge proposals do not presently construct meaning, the project of predictive knowledge is not inherently doomed; quite the opposite, predictive knowledge provides a promising foundation for construction of meaning in Machine Intelligence. We suggest that it may be possible to express generalities by using predictions to construct models of the environment, thereby completing the triadic relation and forming a semiotic process.

References

Barbieri, M. (2007). *Introduction to Biosemiotics: The New Biological Synthesis*. Springer Science & Business Media.

Edwards, A. L., Hebert, J. S., and Pilarski, P. M. (2016). Machine learning and unlearning to autonomously switch between the functions of a myoelectric arm. In *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference On*, pages 514–521. IEEE.

Günther, J., Pilarski, P. M., Helfrich, G., Shen, H., and Diepold, K. (2016). Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning. *Mechatronics*, 34:1–11.

Koop, A. (2008). *Investigating Experience: Temporal Coherence and Empirical Knowledge Representation*. PhD Thesis, University of Alberta.

Modayil, J. and Sutton, R. S. (2014). Prediction driven behavior: Learning predictions that drive fixed responses. In *The AAAI-14 Workshop on Artificial Intelligence and Robotics, Quebec City, Quebec, Canada*.

Peirce, C. S., Hartshorne, C., and Weiss, P. (1931). *Collected Papers of Charles Sanders Peirce. Vol. 1, Principles of Philosophy*. Belknap Press of Harvard University Press.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

Sutton, R. S. (2009). The grand challenge of predictive empirical abstract knowledge. In *Working Notes of the IJCAI-09 Workshop on Grand Challenges for Reasoning from Experiences*.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *AAMAS 2011*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems.

White, A. (2015). *Developing a Predictive Approach to Knowledge*. PhD Thesis, University of Alberta.

When is a Prediction Knowledge?

Alex Kearney
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
kearney@ualberta.ca*

Patrick M. Pilarski
Departments of Medicine and Computing Science
University of Alberta
Edmonton, Alberta, Canada
pilarski@ualberta.ca

Abstract

Within Reinforcement Learning, there is a growing collection of research which aims to express all of an agent's knowledge of the world through predictions about sensation, behaviour, and time. This work can be seen not only as a collection of architectural proposals, but also as the beginnings of a theory of machine knowledge in reinforcement learning. Recent work has expanded what can be expressed using predictions, and developed applications which use predictions to inform decision-making on a variety of synthetic and real-world problems. While promising, we here suggest that the notion of predictions as knowledge in reinforcement learning is as yet underdeveloped: some work explicitly refers to predictions as knowledge, what the requirements are for considering a prediction to be knowledge have yet to be well explored. This specification of the necessary and sufficient conditions of knowledge is important; even if claims about the nature of knowledge are left implicit in technical proposals, the underlying assumptions of such claims have consequences for the systems we design. These consequences manifest in both the way we choose to structure predictive knowledge architectures, and how we evaluate them. In this paper, we take a first step to formalizing predictive knowledge by discussing the relationship of predictive knowledge learning methods to existing theories of knowledge in epistemology. Specifically, we explore the relationships between Generalized Value Functions and epistemic notions of Justification and Truth.

Keywords: Reinforcement Learning, Predictive Knowledge, Continual Learning, General Value Functions, Epistemology

Acknowledgements

Thanks to David Quail for insightful discussion; thanks to Dylan Jones, Kory Mathewson, and Johannes Günther for feedback on an early draft of this manuscript. This work was supported in part by the Canada Research Chairs program, the Alberta Machine Intelligence Institute, Alberta Innovates, the Natural Sciences and Engineering Research Council, and by Borealis AI through their Global Fellowship Award.

*For additional discussion on the epistemology of predictive knowledge and how we can view predictive knowledge architectures as having semantics, please refer to Kearney & Oxton, "Making Meaning: Semiotics Within Predictive Knowledge Architectures", RLDM 2019.

1 Predictive Approaches to Machine Knowledge

One of the foundational goals of machine intelligence is to create systems which are able to understand and reason about the world around them. Within Reinforcement Learning, there is a growing collection of research which attempts to describe the world in terms of predictions about the environment, sometimes called *Predictive Knowledge* (Sutton, 2009; Koop, 2008; Sutton et al., 2011; White, 2015). Predictive knowledge agents describe the world by making many predictions with respect to their behaviour. These predictions can then be interrelated to express more abstract, conceptual aspects of the environment (Schapire and Rivest, 1988). For instance, using a General Value Function, a system could predict whether there is an obstacle to the left or right. Key to this approach is that all predictions—from immediate sensorimotor anticipation, to abstract conceptual expressions of the environment—are described exclusively in terms of sensation, behaviour, and time. As a result of these constraints, predictive knowledge centres itself around methods which are able to construct their own categories, properties and relationships: predictive knowledge is liberated from the process of labelling. This body of work can be seen as not just a collection of engineering proposals, but also as a fledgling approach to describing knowledge from a machine intelligence perspective—as a starting point for applying Epistemology to Reinforcement Learning.

Predictive knowledge methods show promise; however it is unclear to what extent predictions can be considered knowledge. While prediction's special status as knowledge has been alluded to in RL (Sutton et al., 2011; White, 2015), there has been no discussion of the necessary and sufficient conditions for predictions to be considered knowledge, or the assumptions required and consequences which follow from considering predictions to be knowledge. This is more than simply an absence of conceptual discussion in a purely technical endeavour; there are practical challenges to developing predictive knowledge architectures which are particularly pernicious due to a limited understanding of the requirements of knowledge—i.e., how to choose *what* to predict and *how* to predict it independent of designer intervention is largely unknown. Although predictions have proven to be practically useful in reactive control systems in bionic limbs (Edwards et al., 2016) and industrial laser welding (Günther et al., 2016), in each of these instances the predictions learnt by the system and how they are used to inform decison-making is hand-specified by engineers and designers. These problems, at least in part, are a consequence of a poor understanding of the requirements of knowledge.

When we propose that predictions can be interpreted as knowledge, we are making a claim about what knowledge *is*. In this paper, we begin the project of formalizing a theory of knowledge in reinforcement learning by exploring justification and truth in predictive knowledge. Specifically, we 1) highlight evaluation concerns in predictive knowledge architectures, emphasizing how they relate to existing real-world applications; and 2) argue that epistemology is relevant to predictive knowledge research—that epistemology deserves greater attention when designing predictive knowledge architectures. To do so, we examine one of the most fundamental components of predictive knowledge proposals: General Value Functions (GVFs).

2 General Value Functions

When we discuss the requirements of knowledge, it is natural for us to begin by examining how predictive knowledge learning methods relate to formal theories of knowledge. One of the central methods of specifying predictions in predictive knowledge is through General Value Functions. General Value functions estimate the discounted sum of some signal c over discrete time-steps $t = 1, 2, 3, ..., n$ defined as $G_t = \mathbb{E}(\sum_{k=0}^{\infty}(\prod_{j=1}^{k}(\gamma_{t+j}))C_{t+k+1})$. On each time-step the agent receives some vector o_t of observations which describes the environment and takes an action a_t. The observations are used to construct the *agent-state* $\phi : o_t \rightarrow \mathbb{R}^n$: the state of the environment from the agent's perspective. A GVF is parameterized by a set of weights $w \in \mathbb{R}^n$ which when combined with the agent-state produce an estimate of the return $v(s) = w^\top \phi_(o_t)$ The prediction is specified by two sets of parameters: *question parameters* which determine what the prediction is about and *answer parameters* which determine how the prediction is learnt. Question parameters include the signal of interest C, a discounting function dependent on the state of the environment s_t and an action taken a_t, a factor $0 \geq \gamma \geq 1$ which determines how to discount future signals, and a policy π which describes the behaviour over which the predictions are made. Answer parameters include the step-size α which scales updates to the weights, and the eligibility decay λ which determines how much previous states should update their estimates based on the most recent observation. These predictions can be learned online, incrementally using policy evaluation methods such as Temporal-difference learning (Sutton, 1988).

GVFs form a key component of predictive knowledge proposals by acting as the mechanism through which knowledge is constructed(Sutton et al., 2011). Certainly, not all predictions are created equally. Feature construction and amount of experience contribute to the how well the return G_t is estimated. If we center all knowledge as a collection of predictions, how do we evaluate the quality of a predictions as knowledge?

3 Lessons From Epistemology: Barn Facades and Bionic Limbs

Before embarking on determining whether or not accurate predictions can be considered knowledge, it's prudent to have an understanding of what knowledge is. To this end, we introduce arguments from epistemology, the study of knowledge, and ground these arguments in terms of GVFs.

At its core epistemology captures the distinction between systems which *know* that such-and-such is the case and systems which are simply reliably responding to stimuli. While there are many theories that define the necessary and sufficient conditions for knowledge, they can be summarized broadly as requiring Justification, Truth, and Belief (Gettier, 1963). Each of the legs of this tripartite approach to analysing knowledge are meant to constrain what can be admitted as knowledge.

First, one must believe that they have knowledge of something. Belief may seem trivial; however, there are real-world examples of people who are able to complete tasks while not *believing* they are capable of doing so. When blindsighted patients are asked to perform certain visual tasks, they are able to achieve accuracy higher than would be expected by chance, but do not believe their reports are accurate (Humphrey, 2006). A blindsighted person does not assert that they *know* whether or not a stimulus is present; regardless, they are able to complete these tasks with some reliability. Second, the belief must be truthful. Truth separates beliefs which have bearing on the world, and assertions which are incongruous for reality. If someone says they know the moon is made of cheese, we wouldn't say they *know* what the moon is made of, even if they deeply hold this belief. Third, a belief must be justified. Justification serves to separate accidentally true beliefs from those which are right for good reasons; i.e, if you asked someone how to get to the nearest cafe, and their directions happened to be correct, you wouldn't say they were right—you'd say they were *lucky*.

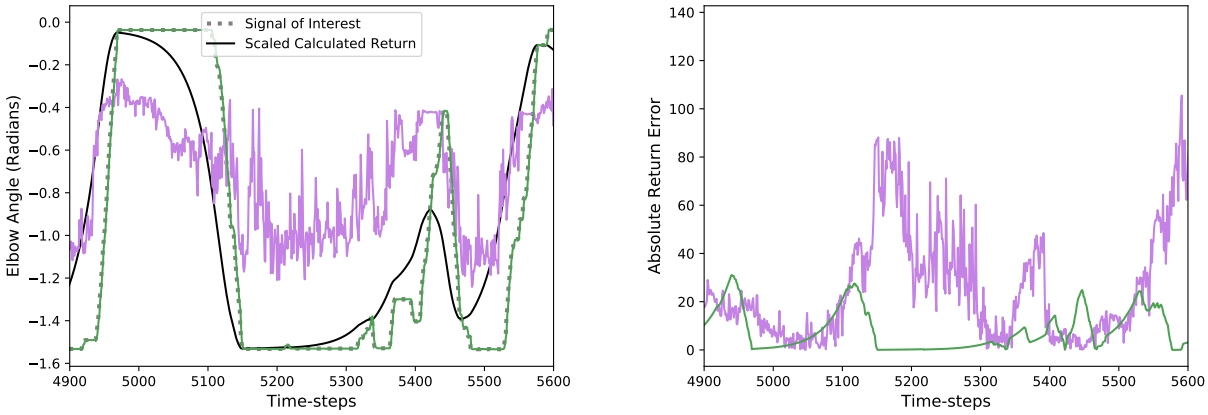
The variety of positions relating to each of justification, truth, and belief are numerous. To that end, we constrain ourselves to considering how GVFs relate to the first two components of the tripod: how can predictions be licensed as being Truthful and Justified? As we alluded to earlier, not all predictions are created equally. In order to make progress in designing predictive architectures, we must be able to separate predictions which are unreliable, or made for poor reasons, from those which are robust and can be used to inform decision-making.

When an agent is making a prediction, it is making an assertion about the world as observed through its data stream. If a prediction is accurate, it is a testament to its truth. One common method of evaluating whether a prediction is correct or not is to compare what is predicted against an estimation of the true return (Pilarski et al., 2012; Edwards et al., 2016; Günther et al., 2016). The approximate return is $\tilde{G}_t = \sum_{k=0}^{b}(\prod_{j=1}^{k}(\gamma_{t+j}))C_{t+k+1} - v_t(s_t)$ for some buffer-size b which determines how many steps into the future cumulants c are stored to produce the return estimate on any given time-step. The truthfulness of the prediction can be described as the the extent to which estimated value matches the true, observed return[1].

If prediction accuracy describes the truth of a prediction, what is justification within predictive knowledge architectures? Or, is justification necessary? As previously mentioned, the necessary and sufficient conditions for knowledge are a point of contention. In the same paper that Gettier introduced Justified True Belief, he argued against its validity. Similarly, Goldman's Barn Facade problem—which we explore in terms of predictions in the following paragraphs—illustrates how evidence and reasons are not the only way to support the claim that a belief is true—reasons are not the only way to separate a lucky guess, from a justified belief (Goldman, 1976). The purpose of justification is simply to show that a belief is expected to be reliable, that a belief is predicted to be true (Brandom, 2009). Can we treat the reliability of predictions as sufficient for identifying knowledge independent of any other form of justification?

In short, no. While the reliability of a belief—or, accuracy of a prediction—is a means of justifying a belief, reliability alone is insufficient to attribute knowledge (Brandom, 2009). We can examine the limitations of reliability as justification by translating Goldman's barn facade problem to a predictive knowledge experiment. Consider a single GVF making a prediction about some signal c. In this case, the return error

[1]This approach is advocated in the original proposal of Sutton et al. (2011)



(a) Estimated value for two predictions; Signal as dotted line.

(b) Absolute return error of two predictions.

Figure 1: Which prediction counts as knowledge: green or purple?

$v_t(\phi_t) - \tilde{G}_t$ is relative to a particular time-step t, and a set of observations o_t. A GVF which predicts random values could make a perfect prediction for a given time-step t and have no return error for an observation o_t. Clearly, the accuracy of a prediction over one time step says nothing about how likely a prediction is to be accurate in general. Given the limitations of a single time-step error, over what horizon—for what period of time, or what collection of states—must we examine the return error to licence truth? Must we calculate the return error of a prediction relative to all possible states in order to determine whether a prediction is sufficiently justified? Such a requirement would be technically infeasible in a real-world setting.

Not only is return error impractical as the exclusive source of justification, it is incoherent on a conceptual level. Relative to each set of states, there is a clear answer as to whether or not a prediction is accurate; however, there is nothing in the world which privileges one set of states over others in making the distinction of truth. So the accuracy, or reliability of a belief does not determine whether or not the prediction is justified. None of this is to say that the reliability of predictions does not have any epistemic significance. Prediction accuracy is unquestionably an important part of assessing the truth of a prediction and evaluating if a prediction is justified. However, Prediction accuracy alone does not tell the full story.

To explore this point, we produce two predictions about the joint angles of a robotic actuator, as sampled from the human controlling a robotic arm to do manipulation task. Please refer to Pilarski et al. (2013a) for the full details of how this dataset was generated. The cumulant of interest is the elbow servo motor angle in radians. For both predictions, the discount factor is $\gamma = 0.99$, corresponding to roughly 2.5 seconds of arm operation. As per Pilarski et al. (2013a), the predictions were made on-policy with TD(λ) with $\lambda = 0.999$ and a step size $\alpha = 0.033$ (Sutton, 1988). Only the function approximators used to construct the agent-state varies between the two predictions.

From the predictions in Figure 1, we can see that the green prediction isn't a prediction at all. Although both predictions are specified to learn the same GVF, the green prediction is simply tracking the signal of interest. In comparison, the purple prediction, in fact, predicts: it rises before the stimulus rises, and decreases before the stimulus falls. Looking at return error alone (Figure 1b), we would be lead to the conclusion that the green prediction is in fact more truthful than the purple. Because the green prediction is more accurate—both on a moment-to moment basis, and throughout the trial—from this *reliabilist* perspective, it is better justified. We could conclude that the green prediction that isn't predicting is a better candidate for knowledge. Although the purple prediction is clearly more predictive, it has a greater return error, both on a moment-to-moment basis on each time-step and in the greater context of the experimental trial.

More than simply a contrived example, these predictions are examples of prototypical GVFs made on bionic limbs to inform control systems. While existing systems are hand-engineered, if we choose to build systems which independently make decisions about what to learn and how to learn them, we must be able to assess the quality of a prediction in a robust, reliable way. From purely an engineering standpoint, in order to build such systems successfully we must be able to discriminate between predictions which have low error for poor reasons and predictions which explain their signal of interest (Pilarski et al., 2013b). Put simply, just because a prediction is accurate, doesn't make it useful.

The limitations of reliability as justification is more than a conceptual problem, it has practical consequences for evaluation in real-world applications of predictive knowledge systems. The consequences of epistemic choices we make—whether we are conscious of them or not—have a fundamental impact on the effectiveness of our systems. To achieve its fullest potential, future work should examine additional methods of supporting the justification of predictions, perhaps using internal signals about learning.

4 Concluding Thoughts: The Importance of Evaluating When Predictions are Knowledge

Within reinforcement learning, there are the seeds of an approach to constructing machine knowledge through prediction. While promising, there is limited discussion of what the formal commitments of such an approach would be: namely, what knowledge is and what counts as it. In this paper, we take a first step towards formalizing predictive knowledge by clarifying the relationship of GVFs to formal theories of knowledge. We identify that a GVF's estimates of some cumulant can be seen as truthful insofar as they match the observed expected discounted return of the cumulant; we discuss arguments for and against the reliability of a belief—or accuracy of a prediction—as being sufficient for justifying knowledge. Having formalized these relationships between GVFs and both justification and truth, we use a robotic prediction task to demonstrate that prediction accuracy is insufficient to determining whether a prediction is knowledge. This inquiry is not simply an academic discussion: it has practical implications for decisions about what knowledge is and what counts as it in architectural proposals. The project of predictive knowledge shows promise not just as a collection of practical engineering proposals, but also as a theory of machine knowledge; however, to achieve its full potential, predictive knowledge research must pay greater attention to the epistemic commitments being made.

References

Brandom, R. (2009). *Articulating Reasons*. Harvard University Press.

Edwards, A. L., Hebert, J. S., and Pilarski, P. M. (2016). Machine learning and unlearning to autonomously switch between the functions of a myoelectric arm. In *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference On*, pages 514–521. IEEE.

Gettier, E. L. (1963). Is justified true belief knowledge? *analysis*, 23(6):121–123.

Goldman, A. I. (1976). Discrimination and perceptual knowledge. *The Journal of Philosophy*, 73(20):771–791.

Günther, J., Pilarski, P. M., Helfrich, G., Shen, H., and Diepold, K. (2016). Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning. *Mechatronics*, 34:1–11.

Humphrey, N. (2006). *Seeing Red*. Harvard University Press.

Koop, A. (2008). *Investigating Experience: Temporal Coherence and Empirical Knowledge Representation*. PhD Thesis, University of Alberta.

Pilarski, P. M., Dawson, M. R., Degris, T., Carey, J. P., Chan, K. M., Hebert, J. S., and Sutton, R. S. (2013a). Adaptive artificial limbs: A real-time approach to prediction and anticipation. *IEEE Robotics & Automation Magazine*, 20(1):53–64.

Pilarski, P. M., Dawson, M. R., Degris, T., Carey, J. P., and Sutton, R. S. (2012). Dynamic switching and real-time machine learning for improved human control of assistive biomedical robots. In *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference On*, pages 296–302. IEEE.

Pilarski, P. M., Dick, T. B., and Sutton, R. S. (2013b). Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints. In *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, pages 1–8. IEEE.

Schapire, R. E. and Rivest, R. L. (1988). *Diversity-Based Inference of Finite Automata*. Master's Thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44.

Sutton, R. S. (2009). The grand challenge of predictive empirical abstract knowledge. In *Working Notes of the IJCAI-09 Workshop on Grand Challenges for Reasoning from Experiences*.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *AAMAS 2011*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems.

White, A. (2015). *Developing a Predictive Approach to Knowledge*. PhD Thesis, University of Alberta.

DeepMellow: Removing the Need for a Target Network in Deep Q-Learning

Seungchan Kim
Department of Computer Science
Brown University
Providence, RI 02906
seungchan_kim@brown.edu

Kavosh Asadi
Department of Computer Science
Brown University
Providence, RI 02906
k8@brown.edu

Michael Littman
Department of Computer Science
Brown University
Providence, RI 02906
mlittman@cs.brown.edu

George Konidaris
Department of Computer Science
Brown University
Providence, RI 02906
gdk@cs.brown.edu

Abstract

Deep Q-Network (DQN) is a learning algorithm that achieves human-level performance in high-dimensional, complex domains like Atari games. One of the important elements in DQN is its use of target network, which is necessary to stabilize learning. We argue that using a target network is incompatible with online reinforcement learning, and it is possible to achieve faster and more stable learning without a target network, when we use an alternative action selection operator, Mellowmax. We present new mathematical properties of Mellowmax, and propose a new algorithm, DeepMellow, which combines DQN and Mellowmax operator. We empirically show that DeepMellow, which does not use a target network, outperforms DQN with a target network.

Keywords: Deep Q-Learning, Deep Q-Network, Target Network

1 Introduction

Deep Q-Network (DQN) [4] is an algorithm that combines standard Q-learning and deep neural networks, which can train agents and yield human-level performances in large-scale, complex domains like Atari games. One of the important techniques used in DQN is a target network, which is a copy of the estimated action-value function that is held fixed to serve as a stable target for some number of steps. However, key shortcomings of target network are that it hinders faster learning by delaying the updates of action-value functions and that it moves us farther from online reinforcement learning, a type of learning long desired by reinforcement learning community[5][7].
We propose an approach that reduces the need for a target network in DQN while ensuring stable learning and good performance in high-dimensional domains. To this end, we use a recently proposed softmax operator, Mellowmax [1]. We derive novel mathematical properties of Mellowmax, and suggest that the use of Mellowmax allows us to remove target network from DQN. We test the performances of a new algorithm, DeepMellow, the combination of Mellowmax and DQN, in two control domains (Acrobot, Lunar Lander) and two Atari games (Breakout, Seaquest). Our empirical results show that DeepMellow achieves more stability than a version of DQN without target network. We also show that DeepMellow, which does not have a target network, learns faster than DQN, which does.

2 Deep Q-Network and Target Network

Deep Q-Network (DQN) is a variation of standard online Q-learning with three modifications: first, it uses deep neural networks to approximate action-value function in large state spaces. Second, it employs an experience replay which stores transition samples into a buffer and randomly samples a minibatch of transitions to update action-value functions. Third, it uses a separate target network, which is just a copy of real action-value function. While the real action-value function is updated continually, the target network is updated with delays to stabilize learning.

The update equation of DQN is as follows:

$$\theta \leftarrow \theta + \alpha\big(r + \gamma \max_{a'} \widehat{Q}(s', a'; \theta^-) - Q(s, a; \theta)\big)\nabla_\theta Q(s, a; \theta),$$

where action-value function Q is parameterized by θ, and a separate target network $\widehat{Q}$ is parameterized by θ^-. The separate weights θ^- is synchronized with θ after some period of time. Using a separate target network makes divergence unlikely, because it adds a delay between the time that $\widehat{Q}$ values are updated and the time that Q values are updated.

We aim to show that it is possible to remove target network from DQN, while ensuring stable learning and good performances in complex domains. There are three reasons for removing the target network from DQN:

(1) Target network in DQN violates online reinforcement learning, and hinders fast learning. Online learning enables real-time learning with streams of incoming data, continually updating the value functions without delays [5][8]. By eliminating the target network, we can remove the delays in the update of value functions, and thus support faster learning, as demonstrated in our experiments. (2) Having a separate target network doubles the memory required to store neural network weights. Thus, removing the target network will contribute to better allocation of memory resources. (3) We aim to develop simpler learning algorithms since they are easier to implement in practice and understand in theory. Target network is an extra complication added to Q-learning to make it work; removing that complication results in a simpler, and therefore, better algorithm. To this end, we use an alternative softmax operator defined next.

3 Properties of Mellowmax Operator

Mellowmax is a softmax operator, which can be thought of as a smooth approximation of the max operator. Softmax operators have been found useful across many disciplines of science, including optimization [2], electrical engineering [9], game theory [10], and experimental psychology [11].

In reinforcement learning, softmax has been used in the context of action selection to trade off exploration (trying new actions) and exploitation (trying good actions), owing to its cheap computational complexity relative to more principled approaches like optimism under uncertainty [12][13] or Bayes optimal decision making [14]. We use softmax in the context of value-function optimization, where we focus on the following softmax operator:

$$mm_\omega(x) := \frac{\log(\frac{1}{n}\sum_{i=1}^{n} \exp(\omega x_i))}{\omega}.$$

This recently introduced operator, called Mellowmax [1], can be incorporated into the Bellman equation as follows:

$$Q(s, a) = \sum_{s' \in S} \mathcal{T}(s, a, s')[R(s, a, s') + \gamma mm_\omega Q(s, \cdot)].$$

Mellowmax has several interesting properties. In addition to being a non-expansion, the parameter ω offers an interpolation between max ($\omega \to \infty$) and mean ($\omega \to 0$) [1]. In the next subsection, we develop two novel mathematical properties of this operator (convexity and monotonic non-decrease).

3.1 Convexity and Monotonic Non-decrease

Claim 1: For any $\omega \geq 0$, $mm_\omega(x)$ is convex.
Proof: Our proof generalizes the proof by Boyd and Vandanberghe [2]. Note that $\nabla^2 mm_\omega(x) = \frac{\omega}{\mathbf{1}^\top z}((\mathbf{1}^\top z)diag(z) - zz^\top)$ where $z_i = e^{\omega x_i}$. They showed that $\frac{1}{\mathbf{1}^\top z}((\mathbf{1}^\top z)diag(z) - zz^\top) \geq 0$. Thus, convexity holds as long as temperature $\omega \geq 0$.

Claim 2: For any $\omega \geq 0$ and any x, $mm_\omega(x)$ is non-decreasing with respect to ω.
Proof: Let $\omega_2 > \omega_1 > 0$. We want to show that $mm_{\omega_2}(x) \geq mm_{\omega_1}(x)$:

$$mm_{\omega_2}(x) = \frac{\log \frac{1}{n}\sum_i e^{\omega_2 x_i}}{\omega_2} = \frac{\log \frac{1}{n}\sum_i e^{\omega_1 x_i \frac{\omega_2}{\omega_1}}}{\omega_2} = \frac{\log \frac{1}{n}\sum_i e^{(\omega_1 x_i)^{(\frac{\omega_2}{\omega_1})}}}{\omega_2}.$$

Using Jensen's Inequality: $\frac{1}{n}\sum_i e^{(\omega_1 x_i)^{(\frac{\omega_2}{\omega_1})}} \geq (\frac{1}{n}\sum_i e^{(\omega_1 x_i)})^{(\frac{\omega_2}{\omega_1})}$. We finally get:

$$mm_{\omega_2}(x) = \frac{\log \frac{1}{n}\sum_i e^{(\omega_1 x_i)^{(\frac{\omega_2}{\omega_1})}}}{\omega_2} \geq \frac{\log(\frac{1}{n}\sum_i e^{(\omega_1 x_i)})^{(\frac{\omega_2}{\omega_1})}}{\omega_2} = \frac{(\frac{\omega_2}{\omega_1})\log(\frac{1}{n}\sum_i e^{(\omega_1 x_i)})}{\omega_2} = \frac{\log(\frac{1}{n}\sum_i e^{(\omega_1 x_i)})}{\omega_1} = mm_{\omega_1}(x),$$

allowing us to conclude that $mm_\omega(x)$ is a non-decreasing function of ω.

4 DeepMellow

Now we present the theoretical basis of DeepMellow algorithm (alleviation of overestimation). Then we compare the differences between DeepMellow and DQN in the next subsection.

4.1 Alleviation of Overestimation

Previous work [6] showed that standard Q-learning, which uses the $\max$ operator, suffers from an overestimation problem: note that due to Jensen's inequality and the convexity of max, $\mathbb{E}[\max \widehat{Q}] \geq \max \mathbb{E}[\widehat{Q}]$. Q-learning can overestimate the target due to noise in the estimator $\widehat{Q}$. In practice, this gap can be quite large. We hypothesize that using a separate target network keeps the target $\widehat{Q}$ constant for a while, and, in effect, removes the randomness from the target. In this case, both sides of the above inequality will be the same quantity $\max \widehat{Q}$.
By the same argument, and as a corollary of the convexity argument of Mellowmax (Claim 1), Q-learning with Mellowmax also suffers from this overestimation problem. However, the magnitude of the overestimation is reduced by lowering the temperature parameter ω, as we argue next. For this analysis, we assume that $\widehat{Q}$ is an unbiased estimate of Q as assumed by previous work. We further assume that $\widehat{Q}$ values are uncorrelated. We wish to find the following gap: $\text{bias}\big(mm_\omega(\widehat{Q})\big) = \mathbb{E}[mm_\omega(\widehat{Q})] - mm_\omega(Q)$. Let's begin with a second-order Taylor expansion of Mellowmax as a good approximation for convex functions: $mm_\omega(y) - mm_\omega(x) \approx \nabla mm_\omega(x)^\top(y-x) + (y-x)^\top \nabla^2 mm_\omega(x)(y-x)$.

Replacing x and y with Q and $\widehat{Q}$, we get: $mm_\omega(\widehat{Q}) - mm_\omega(Q) = \nabla mm_\omega(Q)^\top(\widehat{Q} - Q) + (\widehat{Q} - Q)^\top \nabla^2 mm_\omega(Q)(\widehat{Q} - Q)$.
Taking expectations on both sides, we get: $\mathbb{E}[mm_\omega(\widehat{Q}) - mm_\omega(Q)] = \mathbb{E}[mm_\omega(\widehat{Q})] - mm_\omega(Q) = \text{bias}\big(mm_\omega(\widehat{Q})\big)$.
Thus:

$$\begin{aligned}\text{bias}\big(mm_\omega(\widehat{Q})\big) &= \mathbb{E}[\nabla mm_\omega(Q)^\top(\widehat{Q} - Q)] + \mathbb{E}[(\widehat{Q} - Q)^\top \nabla^2 mm_\omega(Q)(\widehat{Q} - Q)] \\ &= \nabla mm_\omega(Q)^\top \mathbb{E}[\widehat{Q} - Q] + \mathbb{E}[(\widehat{Q} - Q)^\top \nabla^2 mm_\omega(Q)(\widehat{Q} - Q)] \\ = \mathbb{E}[(\widehat{Q} - Q)^\top \nabla^2 mm_\omega(Q) &= \sum_i \frac{\partial^2 mm_\omega(Q)}{\partial (Q_i)^2}\mathbb{E}[(\widehat{Q}_i - Q_i)^2] = \sum_i \frac{\partial^2 mm_\omega(Q)}{\partial (Q_i)^2}\text{Var}[\widehat{Q}_i].\end{aligned}$$

We make two observations about the bias quantity. First, the amount of bias relates to the variance of the estimator. If the estimator $\widehat{Q}$ can perfectly estimate Q with one sample (no variance), then there will also be no bias. Second, note that

$$\frac{\partial^2 mm_\omega(Q)}{\partial (Q_i)^2} = \frac{\omega e^{\omega Q_i}\sum_i e^{\omega Q_i} - w e^{\omega Q_i} e^{\omega Q_i}}{\sum_i e^{\omega Q_i}\sum_i e^{\omega Q_i}} = \omega x - \omega x^2 = \omega x(1-x) \text{ where } w > 0 \text{ and } 0 < x < 1.$$

Here, x denotes $e^{\omega Q_i} / \sum_i \omega Q_i$. We see that the bias is always positive and monotonically increases with ω.

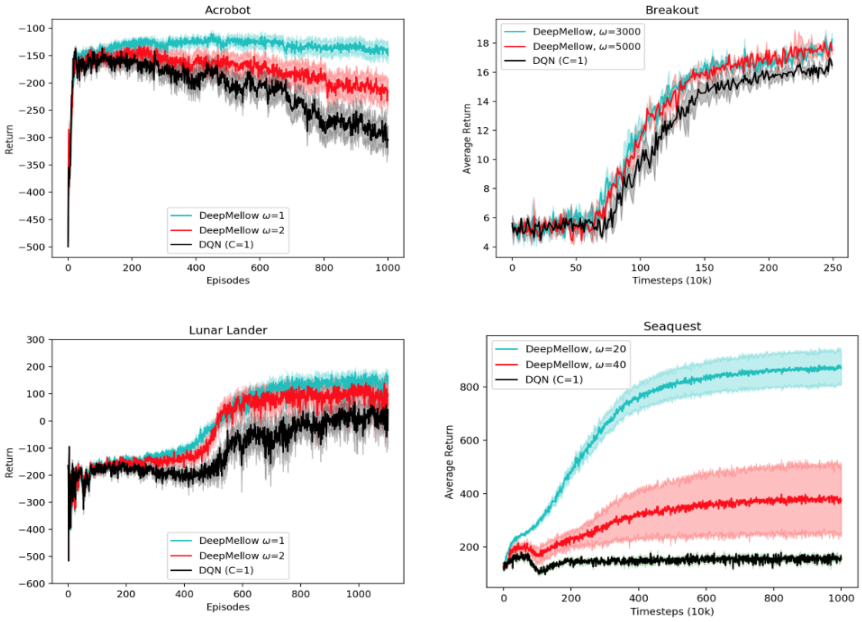


Figure 1: DeepMellow and DQN results with no target network. (Left: Control Domains, Right: Atari Games) DeepMellow outperforms DQN in all domains, in the absence of target network.

4.2 DeepMellow vs DQN

The target network is just a copy of the action-value function that is updated on a delay, and it can serve as a stable target between updates. We note that the analysis in the previous subsection provides an explanation for how a target network improves Q-learning—keeping the target network fixed reduces the variance of estimator $\widehat{Q}$. As we showed above, the variance of the estimator is connected to the amount of bias, so using a target network results in a bias reduction. Our analysis suggests that the use of Mellowmax reduces overestimation bias, and thus reduces the need for a target network.

DeepMellow replaces the max operator in DQN with the Mellowmax operator, as in the framework of generalized MDPs [3]. DeepMellow further differs from DQN, as it does not use a separate target action-value function $\widehat{Q}$, and thus does not need to copy the action-value function every C steps. Therefore, to update action-value function, DeepMellow performs gradient descent on $\{r_j + \gamma mm_\omega \underset{a'}{Q}(\phi_{j+1}, a'; \theta) - Q(\phi_j, a_j; \theta)\}^2$, while DQN performs gradient descent on $\{r_j + \gamma \max_{a'} \widehat{Q}(\phi_{j+1}, a'; \theta^-) - Q(\phi_j, a_j; \theta)\}^2$.

5 Experiments and Results

We tested our DeepMellow algorithm in two control domains (Acrobot, Lunar Lander) and two Atari game domains (Breakout, Seaquest). We first compared DeepMellow and DQN in the absence of a target network. The results are shown in the Figure 1. In Acrobot, DeepMellow achieves more stable learning than DQN—without a target network, the learning curve of DQN goes upward fast, but soon starts fluctuating and fails to improve towards the end. By contrast, DeepMellow (especially with temperature parameter $\omega = 1$) succeeds. Similar results are observed in Lunar Lander. DeepMellow ($\omega \in \{1, 2\}$) achieves more stable learning and higher average returns than DQN without a target network. We could observe similar trends in Atari games, too. In Breakout, DeepMellow ($\omega \in \{3000, 5000\}$) showed better performances than DQN without target network. In Seaquest, the performance gaps widened: DQN without target network couldn't learn, but DeepMellow ($\omega = 20$) was able to learn stably without target network.

Next, to see if DeepMellow has an advantage over DQN with a target network, we compared the performance of the two approaches, focusing on learning speed. As shown in Figure 2, DeepMellow learns faster than DQN in Lunar Lander, Breakout, and Seaquest domains. In Acrobot (not shown), there was no significant difference because both algorithms learned so quickly. In Lunar Lander domain, DeepMellow reaches a score of 0 at episode 517 on average, while DQN reaches the same point around episode 561 on average. Similar results hold in Breakout; DeepMellow ($\omega = 3000$) reaches a score of 12 at timestep 101×10^4, while DQN reaches it at 117×10^4. In Seaquest, DeepMellow reaches to the scores of 400 at timestep 159×10^4; DQN reaches to the same scores at timestep 212×10^4.)

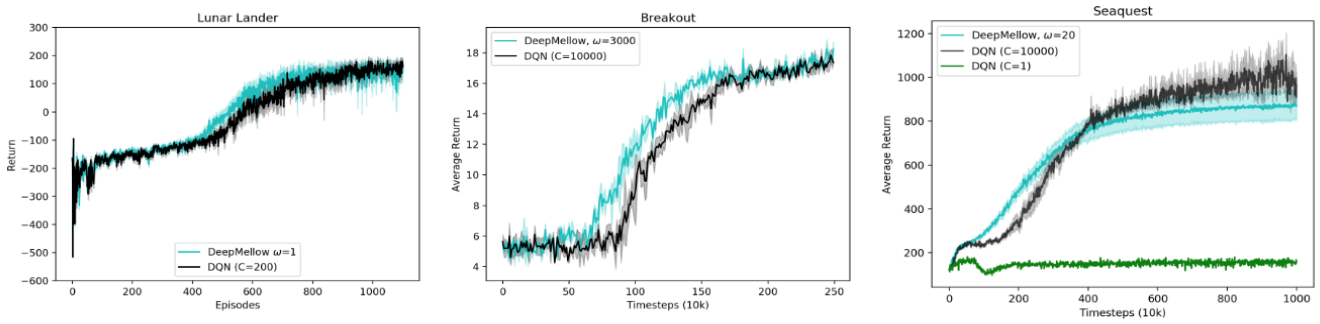


Figure 2: Performances of DeepMellow (with no target network) and DQN (with a target network). DeepMellow has an advantage over DQN in terms of learning speed (how fast it starts learning at the beginning phase of training).

6 Conclusion

We proposed a new algorithm, DeepMellow, that can learn stably without the use of a target network. DeepMellow replaces the max operator in DQN with the Mellowmax operator. We showed new mathematical properties of the Mellowmax operator (convexity, monotonic non-decrease, and mitigation of overestimation) and explained how we can reduce the instability of deep Q-learning using Mellowmax. This increased stability reduces the need for a target network, speeding up learning. Our empirical results show that DeepMellow achieves more stability and higher average scores than DQN without target network. We also showed that DeepMellow without a target network has a learning speed advantage over DQN with a target network.

References

[1] Kavosh Asadi and Michael L.Littman. An alternative softmax operator for reinforcement learning. *In Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pages 243 - 252, 2017.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004

[3] Michael L.Littman and Csaba Szepesvari. A generalized reinforcement-learning model: Convergence and applications. In *ICML*, volume 96, pages 310 - 318, 1996.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G.Bellemare, Alex Graves, Martin A. Riedmiller, AndreasFidjeland, Georg Ostrovski, Stig Petersen, CharlesBeattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533, 2015.

[5] Richard S. Sutton and Andrew G.Barto. Reinforcement learning - an introduction. Adaptive computation and machine learning. MIT Press, 1998.

[6] Hado van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia,Canada*, pages 2613-2621, 2010.

[7] Harm Seijen and Rich Sutton. True online td (lambda). In *International Conference on Machine Learning*, pages 692-700, 2014.

[8] A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994

[9] Aysel Safak. Statistical analysis of the powersum of multiple correlated log-normal components. *IEEE Transactions on Vehicular Technology*, 42(1):5861, 1993

[10] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. arXiv preprint arXiv:1704.00805, 2017.

[11] Dale O Stahl II and Paul W Wilson. Experimental evidence on players models of other players. *Journal of economic behavior & organization*, 25(3):309-327, 1994.

[12] Ronen I. Brafman and Moshe Tennenholtz. R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213-231, 2002.

[13] Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881-888. ACM, 2006.

[14] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. In AAAI/IAAI, pages761-768, 1998

Variational State Encoding as Intrinsic Motivation in Reinforcement Learning

Martin Klissarov
School of Computer Science
McGill University
martin.klissarov@mail.mcgill.ca

Riashat Islam
School of Computer Science
McGill University
riashat.islam@mail.mcgill.ca

Khimya Khetarpal
School of Computer Science
McGill University
khimya.khetarpal@mail.mcgill.ca

Doina Precup
School of Computer Science
McGill University
dprecup@cs.mcgill.ca

Abstract

Discovering efficient exploration strategies is a central challenge in reinforcement learning (RL), especially in the context of sparse rewards environments. We postulate that to discover such strategies, an RL agent should be able to identify surprising, and potentially useful, states where the agent encounters meaningful information that deviates from its prior beliefs of the environment. Intuitively, this approach could be understood as leveraging a measure of an agent's surprise to guide exploration. To this end, we provide a straightforward mechanism by training a variational auto-encoder to extract the latent structure of the task. Importantly, variational auto-encoders maintain a posterior distribution over this latent structure. By measuring the difference between this distribution and the agent's prior beliefs, we are able to identify states which potentially hold meaningful information. Leveraging this as a measure of intrinsic motivation, we empirically demonstrate that an agent can solve a series of challenging sparse reward, highly stochastic and partially observable maze tasks.

Keywords: Intrinsic Motivation, Surprise, Exploration, State Encoding, Variational Auto-Encoders, Sparse Reward,

1 Introduction

Reinforcement learning (RL) algorithms have achieved several recent accomplishments, especially by using non-linear function approximators to solve high dimensional complex tasks. However, most RL algorithms rely on a well designed reward functions to guide the behaviour of the agent. Hand-crafting such reward functions is complex and can sometimes lead to unexpected behaviour. In order to be deployed in real-world settings, RL agents will have to be able to learn from sparse rewards environments. A key step towards scaling RL algorithms for unknown reward functions is for the agent to naturally adapt its behaviour by learning a good exploration strategy.

Exploiting task structure is a key step towards learning efficient exploration strategies in RL. Recent approaches include the discovery of bottleneck states [Goyal et al.2019] or learning a feature space [François-Lavet et al.2018]. Exploration can also be formulated of as an agent's internal drive towards learning more about the environment. This is often defined as *intrinsic motivation*, or curiosity of the agent [Schmidhuber1991a, Oudeyer et al.2016]. Intrinsic motivation is also an important concept in developmental psychology, where it is defined as the desire to pursue an activity for its inherent satisfaction rather than for some external pressure or reward [Oudeyer and Kaplan2009]. Curiosity or intrinsic motivation can therefore be thought of as a task agnostic exploration heuristic towards the goal of learning in an online fashion based on the agent's interactions with the environment.

In this work, we propose a formulation of intrinsic motivation based on the definition of Bayesian surprise [Itti and Baldi2009]. The intuition behind this approach is that experiences which deviate from the agent's prior beliefs about the world are surprising, and potentially useful for learning. In other words, the agent should be able to identify the states which create important changes to its prior knowledge by measuring the difference between posterior and prior distribution after visiting such states. We propose a framework to identify surprising or useful states in the environment via latent representation learning, which we use as intrinsic motivation for solving sparse rewards and partially observable maze tasks.

Our Contributions : We use a Variational Auto-Encoder (VAE) to project the state space into a probabilistic latent representation that would represent the inherent structure of the environment. By using a VAE we naturally obtain a measure of the agent's surprise defined by how much the posterior distribution over the latent representation deviates from its prior belief. This is measured in the form of a KL divergence $KL(p(Z|S)||p(Z))$ where $p(Z)$ is the agent's prior distribution over the latent structure of the environment and $p(Z|S)$ the posterior. We incentivize the agent to visit surprising (and potentially useful) regions of the state space by providing this KL divergence as intrinsic motivation.

2 Preliminaries and Background

In this work we will consider the standard reinforcement learning setting which considers the environment as a Markov Decision Process $\mathcal{M}$, which is defined as a tuple $\doteq(\mathcal{S}, \mathcal{A}, \gamma, r, P)$. $\mathcal{S}$ is the state set, $\mathcal{A}$ the action set, $\gamma \in [0, 1)$ the discount factor, $r : S \times A \rightarrow Dist(\mathbb{R})$ the reward function and $P : S \times A \rightarrow Dist(S)$ the transition probability distribution. A policy $\pi : S \rightarrow Dist(A)$ specifies a way of behaving, and its value function is the expected return obtained by following π: $V_\pi(s) \doteq \mathbb{E}_\pi \left[\sum_{t=0} \gamma^t r(S_t, A_t) | S_0 = s\right]$. V_π satisfies the following Bellman equations: $V_\pi(s) = \sum_a \pi(a|s) \left(r(s, a) + \gamma \sum_{s'} P(s'|s, a) V_\pi(s')\right)$.

Curiosity as a form of intrinsic motivation has been argued to be a fundamental component for efficient learning [Friston et al.2006]. One of the ways to implement curiosity is by maintaining a forward dynamics model of the environment and using its prediction error [Pathak et al.2017, Schmidhuber1991b] or prediction uncertainty [Houthooft et al.2016] as intrinsic reward. These approaches encourage the agent to visit regions of the state space where the dynamics of the environment are less well understood, therefore guiding exploration. However, their performance tends to suffer in stochastic environments as it becomes harder to predict the consequences of the agent's actions. Another direction [Ostrovski et al.2017] formulated an exploration bonus as a measure of novelty in terms of unseen states. Such approaches have shown great potential but contain some strict requirements on the density model of the states, such as it should be learning-positive. Other ways to improve exploration include the optimal rewards framework [Singh et al.2010] where the authors propose that the optimal intrinsic reward is the one that would maximize the extrinsic reward. However, defining such optimal reward function is an open question. Our approach on the other hand aligns with the Bayesian perspective on surprise [Itti and Baldi2009] which has been shown to be applicable across different spatio-temporal scales and levels of abstractions.

3 Leveraging State Encoding for Intrinsic Motivation

3.1 Intrinsic motivation

In this work, we assume that the experiences S of an agent are generated by a random latent process defined through the variable Z. This latent process can encode some structure or pattern present in the observed data. The goal of

the agent would be to extract and learn this structure in order to come to a better understanding of the world it is interacting with. However, to faithfully represent the latent factors of variation, an agent has to successfully explore the environment. Therefore, the objective of extracting structure from the environment is deeply interlaced with the objective of exploration. One way to attend to this challenge is by adding an intrinsic reward that would depend on the quality of the model of the environment. On one hand, this intrinsic motivation would encourage the agent to gather unseen data which would improve the model, while on the other hand guiding the agent to fully explore its environment.

We propose a measure of intrinsic motivation formulated as the distance between the posterior distribution over the latent variable $p(Z|S)$ after seeing new data S and the prior $p(Z)$. A natural way to measure this distance is through the KL divergence. Therefore, we can define the intrinsic reward at a state S as

$$r_{intrinsic}(S) = KL\big((p(Z|S)||p(Z)\big)$$

Our measure of intrinsic reward is closely related to the definition of Bayesian surprise proposed by [Itti and Baldi2009]. In this work, the authors argue that the only rigorous definition of surprise is by measuring how data affects the beliefs of an observer about the world. This measure of surprise is done by computing the difference between the prior distribution $p(M)$ of the observer, where M represents the possible models of its environment, and posterior distribution $p(M|D)$ after observing data D. Our definition of intrinsic motivation can then be seen as an approximation to Bayesian surprise, with the slight conceptual difference that the variable Z represents a latent encoding of the structure of the environment. This difference will have an impact in our work as it directly guides our implementation.

3.2 Approach

It is usually impractical to infer exactly the posterior distribution $p(Z|S)$ as it involves intractable integrals. We will therefore choose to approximate this posterior by a variational distribution $q_\phi(Z|S)$. A natural candidate to represent this distribution is through a Variational Auto-Encoder (VAE). VAEs take the inputs S and project them into latent space Z, which is usually of smaller dimensionality. This latent space is meant to capture factors of variation (patterns) within the data. Importantly, a VAE minimizes the following loss:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(Z|S)}\big[\log p_\theta(S|Z)\big] - KL\big(q_\phi(Z|S)||p(Z)\big)$$

where the first term is the reconstruction loss while the second term encourages the approximate posterior $q_\phi(Z|S)$ to stay close to the prior $p(Z)$. In practice, the prior is chosen as a unit Gaussian to simplify the implementation. This also our choice for the prior.

Algorithm 1: Training loop with intrinsic motivation for A2C.

for *Episode=0,1,2,....* **do**
  Initialize dataset $\mathcal{D}$ and insert s_0 in $\mathcal{D}$.
  for *t=0,1,2...T* **do**
    Take action a_t and observe next state s_{t+1} and extrinsic reward $r_{extrinsic}(s_{t+1})$
    Compute intrinsic reward: $r_{intrinsic}(s_{t+1}) = KL(q_\phi(z|s_{t+1})||p(z))$
    Store tuple $(s_{t+1}, a_t, r_{intrinsic}(s_{t+1}), r_{extrinsic}(s_{t+1}))$ in $\mathcal{D}$
    if *mod(t,N)* **then**
      Train the actor and critic on return $G_t = \sum_t r_{extrinsic}(s_t) + \beta KL(q_\phi(z|s_t)||p(z))$
      Train the VAE on the collected states s in $\mathcal{D}$.
      Initialize dataset $\mathcal{D}$ and insert s_t in $\mathcal{D}$.
  end
end

The overall loss function is a lower-bound to the likelihood of the data. This lower-bound is appealing as it explicitly evaluates the KL divergence between posterior and prior distributions. It is therefore straightforward to leverage VAEs for intrinsic motivation. To do so, we need to separately train a VAE on the stream of data an RL agent experiences. We can then define the useful states, or states which contain a high degree of surprise, in places where the KL is high between the posterior and the prior. This KL between the posterior and prior, whenever high, would encourage the agent to visit that region of the state space when it is provided as intrinsic motivation. By doing so, the agent would efficiently explore its environment and improve the quality of the VAE for encoding the hidden structure in the data.

We define the intrinsic motivation reward as $r_{intrinsic}(s_t) = KL(q_\phi(z|s_t)||p(z))$ such that at every step, the agent gets a total reward of $r_{total}(s_t) = r_{extrinsic}(s_t) + \beta r_{intrinsic}(s_t)$. We can therefore define policy gradient objectives based on the cumulative discounted total return, which includes both the extrinsic and intrinsic task rewards. In our implementation, we use will be using actor-critic to solve the task at hand. However, our definition of intrinsic motivation could be readily used with any other policy gradient algorithm, as well as value-based algorithms. We provide a description of the overall process in Algorithm 1.

4 Experimental Results

We will perform experiments on multi-room maze tasks, which are partially observable and sparse reward tasks, as part of the MiniGrid environment [Chevalier-Boisvert and Willems2018]. In these environments, the agent has to navigate a number of rooms, by opening doors or by using a key, in order to get to the goal situated at the other end of the maze. Due to the sparsity in rewards, these maze tasks are often hard to solve, hence requiring efficient exploration strategies. The goal of our experiments is to show that our definition of intrinsic motivation can achieve efficient exploration. To do so, we compare our implementation (VAE) to two baselines: a standard A2C agent and an A2C agent using the prediction error of a model of the transition dynamics as intrinsic motivation (ICM) [Pathak et al.2017]. In Figure 1 we show empirical results on three domains: Multi-Room-N3S4, Multi-Room-N4S4 (where N represents the number of rooms and S the size of the rooms) and Door-Key-8x8. We see that our approach, VAE, and the approach based on the prediction error, ICM, both outperform significantly the A2C baseline. In the Multi-Room-N4S4 environment, we notice that our approach outperforms ICM. Upon investigating the behaviour of the agent, we noticed that the KL divergence was highest at key states such as hallways, in the sight of the door and near the goal. Therefore, we believe that one of the reasons why our agent seems to perform better is due to a possible correlation between surprising states and useful states in these particular environments.

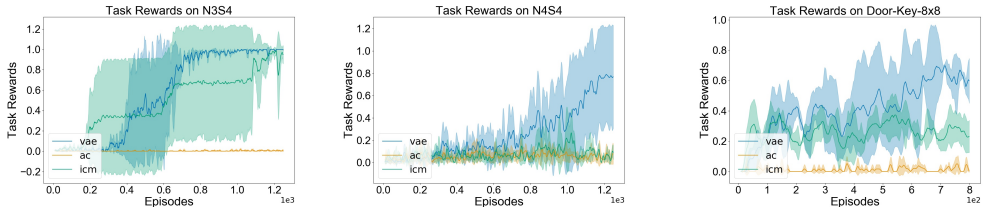
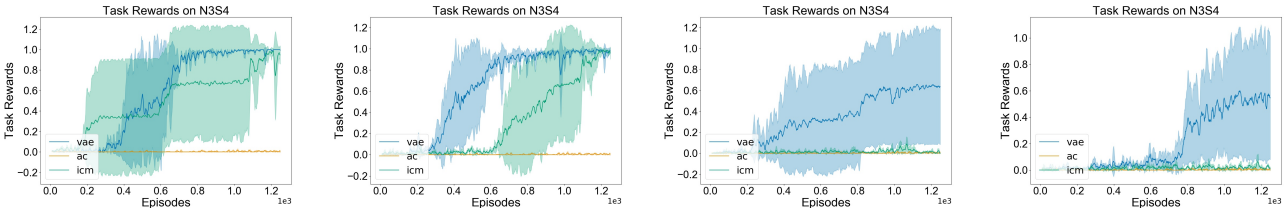


Figure 1: **Task Rewards** on partially observable and sparse reward tasks from the MiniGrid environment. We see that our approach, *VAE, significantly outperforms* both the approach based on the prediction error i.e. ICM, and the A2C baseline.

It is widely-known that intrinsic motivation based on the prediction error of a transition model is sensitive to the inherent stochasticity of the environment [Burda et al.2018]. As such, we performed a series of experiments on the same task but with different degrees of randomness and we show our results in Figure 2. We notice that as the stochasticity in the environment is increased (from left to right), the prediction error of ICM becomes an unreliable source of intrinsic rewards which in turn degrades the performance of the agent on the task. This highlights an important difference with our approach: the agent is not trying to predict the consequences of its actions, as sometimes they can be very complex, but instead tries to encode the structure present in the stream of observations. This provides a efficient intrinsic signal that can guide the agent even when the environment becomes less predictable.



(a) (b) (c) (d)

Figure 2: **Task Rewards for different degrees of environmental stochasticity**: As the stochasticity in the environment is increased from (a) to (d), the prediction error of ICM becomes an unreliable source of intrinsic rewards which in turn degrades the performance of the agent on the task. On the contrary, our approach VAE shows consistency and is robust to stochasticity in the environment.

5 Discussion and Future Work

In this work, we presented an interestingly simple approach towards intrinsic motivation inspired by the definition of Bayesian surprise. We emphasize that our approach is readily extendable towards existing RL frameworks as it requires little overhead. In contrast to several existing works which use prediction error of a transition dynamics model as intrinsic motivation, our approach does not suffer much from an increase in the environment's stochasticity.

A possible improvement to the current framework would be to use a model of the environment that better reflect its latent structure. As it has been noted in [Ha and Schmidhuber2018], variational auto-encoders tend to encode details about the observations that are not always meaningful. To overcome this issue, we could use multiple losses to refine the latent representation [François-Lavet et al.2018].

The key to our approach is to provide intrinsic motivation that is only dependant on the latent structure of the environment. This means that, irrespective of a dense or sparse reward environments, we can provide an exploration bonus defined by the agent's measure of Bayesian surprise, without requiring the agent to know the task-dependent goal information. This is an interesting step towards transfer learning, where even if the task reward changes or the transition dynamics change, the agent can use the learnt state encoding representation as intrinsic motivation in new tasks. In future work, we aim to evaluate the usefulness of our proposed method on transfer learning tasks where we would provide an exploration bonus in new tasks with the same learnt variational encoder.

References

[Burda et al.2018] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*.

[Chevalier-Boisvert and Willems2018] Chevalier-Boisvert, M. and Willems, L. (2018). Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid.

[François-Lavet et al.2018] François-Lavet, V., Bengio, Y., Precup, D., and Pineau, J. (2018). Combined reinforcement learning via abstract representations. *CoRR*, abs/1809.04506.

[Friston et al.2006] Friston, K., Kilner, J., and Harrison, L. (2006). A free energy principle for the brain. *Journal of Physiology-Paris*, 100(1-3):70–87.

[Goyal et al.2019] Goyal, A., Islam, R., Strouse, D., Ahmed, Z., Botvinick, M., Larochelle, H., Levine, S., and Bengio, Y. (2019). Infobot: Transfer and exploration via the information bottleneck. *CoRR*, abs/1901.10902.

[Ha and Schmidhuber2018] Ha, D. and Schmidhuber, J. (2018). World models. *CoRR*, abs/1803.10122.

[Houthooft et al.2016] Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117.

[Itti and Baldi2009] Itti, L. and Baldi, P. (2009). Bayesian surprise attracts human attention. *Vision research*, 49(10):1295–1306.

[Ostrovski et al.2017] Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. (2017). Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2721–2730. JMLR. org.

[Oudeyer et al.2016] Oudeyer, P.-Y., Gottlieb, J., and Lopes, M. (2016). Intrinsic motivation, curiosity, and learning: Theory and applications in educational technologies. In *Progress in brain research*, volume 229, pages 257–284. Elsevier.

[Oudeyer and Kaplan2009] Oudeyer, P.-Y. and Kaplan, F. (2009). What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6.

[Pathak et al.2017] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17.

[Schmidhuber1991a] Schmidhuber, J. (1991a). Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE.

[Schmidhuber1991b] Schmidhuber, J. (1991b). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227.

[Singh et al.2010] Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. (2010). Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82.

Graph-DQN: Fast generalization to novel objects using prior relational knowledge

Varun V. Kumar
Intel AI Lab
varun.v.kumar@intel.com

Hanlin Tang
Intel AI Lab
hanlin.tang@intel.com

Arjun K. Bansal
Intel AI Lab
arjun.bansal@intel.com

Abstract

Humans have a remarkable ability to both generalize known actions to novel objects, and reason about novel objects once their relationship to known objects is understood. For example, on being told a novel object (e.g. 'bees') is to be avoided, we readily apply our prior experience avoiding known objects without needing to experience a sting. Deep Reinforcement Learning (RL) has achieved many remarkable successes in recent years including results with Atari [3] games and Go [2] that have matched or exceeded human performance. While a human playing Atari games can, with a few sentences of natural language instruction, quickly reach a decent level of performance, modern end-to-end deep reinforcement learning methods still require millions of frames of experience. Past studies have hypothesized a role for prior knowledge in addressing this gap between human performance and Deep RL [5]. However, scalable approaches for combining prior or instructional knowledge with deep reinforcement learning have remained elusive.

We introduce a graph convolution based reinforcement learning architecture (Graph-DQN) for combining prior information, structured as a knowledge graph, with the visual scene, and demonstrate that this approach is able to generalize to novel objects whereas the baseline algorithms fail. Ablation experiments show that the agents apply learned self-object relationships to novel objects at test time. In both a Warehouse game and the more complex Pacman environment, Graph-DQN is also more sample efficient, reaching the same performance in 5-10x fewer episodes compared to the baseline. Once the Graph-DQN is trained, we can manipulate agent behavior by modifying the knowledge graph in semantically meaningful ways. These results suggest that Graph-DQNs provide a framework for agents to reason over structured knowledge graphs while still leveraging gradient based learning approaches.

1 Graph-DQN

In deep learning based RL agents such as Deep Q-Networks (DQN), the input state is a 2-D feature map representing the game world as either RGB pixels, or as a symbolic environment. In this paper, we design Graph-DQNs for symbolic grid worlds. Our model combines a provided prior knowledge graph with the symbolic environment, and leverages graph convolutions to reason over entities in both the knowledge graph and the game state.

1.1 Knowledge Graph

The knowledge graph $\mathcal{K} = (\mathcal{V}, \mathcal{E})$ is a directed graph provided as vertices for each symbol in the environment (for subjects and objects), $\mathcal{V} = \{v_A, v_b, v_B, v_+, \ldots\}$ initially encoded as one-hot vectors of length $|\mathcal{V}|$, and edge features $\mathcal{E} = \{e_{Ab}, e_{AB}, e_{A+}, \ldots\}$. The edge features (for relations) are represented as one-hot vectors. The connectivity of the graph, as well as the edge features are designed to reflect the structure of the environment. During training, the knowledge graph's structure and features are fixed. Importantly, while we provide the one-hot encoded representation of the edge relationships, the Graph-DQN must learn to ground the meaning of this representation in terms of rewarding actions during training. I

1.2 Graph DQN model

The first component of our graph DQN model performs $\mathcal{K} \rightarrow \mathcal{S}'$ and infuses the knowledge graph into the game state as follows:

$$\mathcal{K} \rightarrow \texttt{GraphConv} \rightarrow \texttt{GraphConv} \rightarrow \mathcal{K}' \rightarrow \texttt{Broadcast}(\mathcal{S}, \mathcal{K}') \rightarrow \mathcal{S}' \tag{1}$$

The features of the nodes in the knowledge graph $\mathcal{K}$ are enriched through the use of edge-conditioned graph convolution [7]. The enriched graph $\mathcal{K}'$ are then `Broadcast` into the state S by copying the node features into the locations in S of the corresponding symbols, producing a game state S′.

The second component transfers information from $\mathcal{S}' \rightarrow \mathcal{K}$ with:

$$(\mathcal{S}', \mathcal{K}') \rightarrow \texttt{KGConv} \rightarrow \mathcal{S}'' \rightarrow \texttt{Pooling} \rightarrow \mathcal{K} \rightarrow \texttt{GraphConv} \rightarrow \texttt{GraphConv} \rightarrow \mathcal{K}''' \tag{2}$$

Here we apply a joint convolution over both S′ and K′ computed in the first component. The result is then pooled into the knowledge graph by averaging over all the features from the state corresponding to the symbol location. We then apply two layers of graph convolution to produce the updated knowledge graph representation $\mathcal{K}'''$.

Finally, the network predicts the Q-value by

$$(\mathcal{S}''', \mathcal{K}''') \rightarrow \texttt{KGConv} \rightarrow \mathit{Dense}(50) \rightarrow \mathit{Dense}(4) \tag{3}$$

Through end to end training, the network learns to reason over both the knowledge graph and the state representations, and uses `Pooling`, `Broadcast`, and `KGConv` to communicate between the two representation types. As shown in the experiments below, the model learned to ground the knowledge graph representation in terms of actions, and used this representation during the test phase when it encountered novel objects connected with known relationships to entities in the knowledge graph.

2 Experiments

Previous environments measured generalization to more difficult levels [1], modified environment dynamics [6], or different solution paths [9]. These environments, however, do not introduce new objects at test time. To quantify the generalization ability of Graph-DQN to unseen objects, we needed a symbolic game with the ability to increment the difficulty in terms of the number of new objects and relationships. Therefore, we introduce a new Warehouse environment, where the agent pushed balls into the corresponding bucket, and new ball and bucket objects and their pairing are provided at test time. We also benchmarked our model and the baseline DQN algorithm on a symbolic version of Pacman[1].

[1] `http://ai.berkeley.edu/project_overview.html`

Table 1: Experiment variations for the Warehouse environment. The agent is rewarded for pushing the ball into the correct bucket. For each type, we list the rewarded ball-bucket pairs in the training and test games. Note that the test games only include ball types not seen in the training games. Sets denote rewarded combinations. For example, $\{b, c\} \rightarrow B$ means $b \rightarrow B$ and $c \rightarrow B$ are rewarded.

Name	Training Pairs	Test Pairs
one-one	$b \rightarrow B$	$c \rightarrow B$
two-one	$\{b, c\} \rightarrow B$	$d \rightarrow B$
five-two	$\{b, c, d, e, f\} \rightarrow B$	$\{g, h\} \rightarrow B$
buckets	$b \rightarrow B$, $c \rightarrow C$, $d \rightarrow D$, $e \rightarrow E$, $f \rightarrow F$	$g \rightarrow G$, $h \rightarrow H$, $i \rightarrow I$, $j \rightarrow J$, $k \rightarrow K$
buckets-repeat	$\{b, c, d\} \rightarrow B$, $\{e, f, g\} \rightarrow C, \ldots,$ $\{n, o, p\} \rightarrow F$	$\{q, r, s\} \rightarrow G$, $\{t, u, v\} \rightarrow H$, $\ldots, \{6, 7, 8\} \rightarrow K$

2.1 Warehouse

The warehouse environment is implemented using the *pycolab* environment [8]. The environment consists of a 10×10 grid, where the agent is rewarded for pushing balls into their matching buckets. The set of rewarded ball-bucket pairs varies, and in the test games the agent sees balls or buckets not seen during training. For the variations, see Table 1. Lower case alphanumeric characters refer to balls, and upper case as buckets. We increasingly vary the difficulty of the environment by the number of ball-bucket pairs, the complexity of the grouping, and the number of unseen objects. The buckets-repeat is a challenging environment, with complex relationships in the test environment. The agent is identified as the `A` symbol, and the walls with `+`.

2.2 Symbolic Pacman

We test the agents on the smallGrid, mediumGrid, mediumClassic, and capsuleClassic environments from the text-based Pacman implementation. The environments differed in the size of the map as well as the numbers of ghosts, coins, and capsules present. We used random ghosts (as opposed to ghosts seeking out the agent).

2.3 Knowledge graph construction

For both environments, we add all entities to the knowledge graph with the exception of blank spaces. We then add edges between objects to reflect relationships present in the game structure. Each entity or edge type is assigned a unique one-hot vector; note however that edges between two pairs of entities may have the same edge type if they are connected with a similar relationship.

3 Results

We compared our Graph-DQN model with the baseline DQN in the Warehouse and Pacman environments. In addition, we compared the performance of different knowledge graph architectures during training. We also demonstrated the ability to manipulate agent behavior by changing the knowledge graph at test time.

3.1 Warehouse

In the Warehouse environment, the Graph-DQN model was more sample efficient during training than the baseline Conv-DQN algorithm, as shown in Figure 1. For example, in the one-one environment, our model required approximately 8x fewer samples to reach the solution in the training environment (compare blue and green curves in the top row). In addition, in more complex environments with an increased number of possible objects and ball-bucket pairings, the baseline Conv-DQN required increasingly more samples to solve, whereas the Graph-DQN solved in the same number of samples.

We tested zero-shot transfer learning by placing the trained agents in environments with objects unseen during training. The Graph-DQN is able to leverage the knowledge graph to generalize, solving in $> 80\%$ of the test environments (see Figure 1, bottom row). The baseline DQN failed completely to generalize to these environments. Additional control experiments ruled out confounding factors.

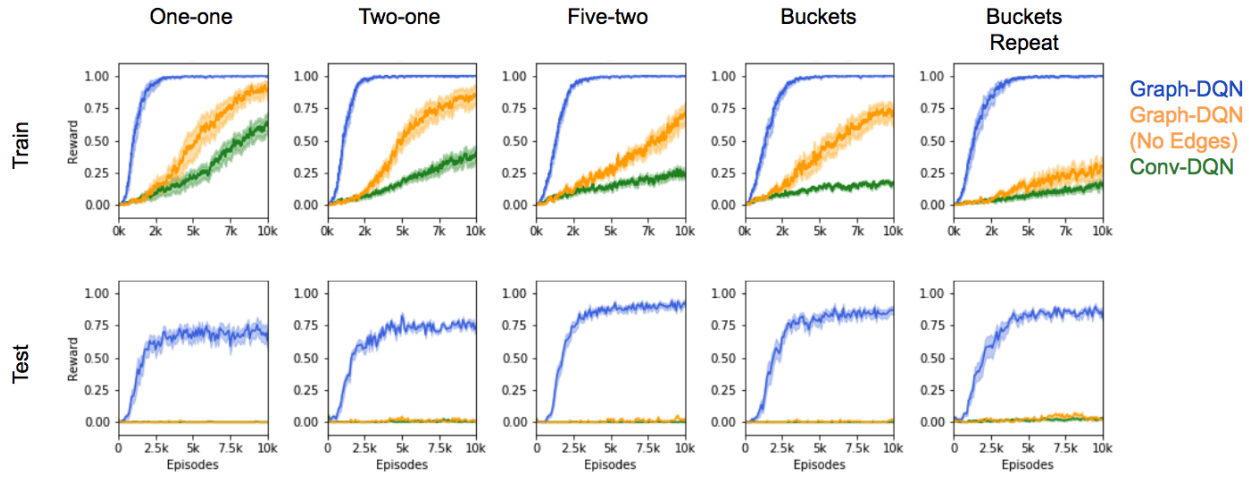


Figure 1: Warehouse results. For the environments described in Table 1 (columns), performance of the baseline DQN (green), our proposed Graph-DQN (blue), and a variant of Graph-DQN with edges removed (orange) over the number of training episodes. The top row represents the success rate (fraction of environments completed within 100 steps) in the training environments, and bottom row measures success rate on the test environments. Bold lines are the mean success rate over $n = 10$ runs, and shaded area denotes the standard error. A moving average of $t = 100$ episodes was applied. The Graph-DQN model is more sample efficient during training, and also generalizes to test environments.

3.2 Pacman

We compare Graph-DQN to the baseline Conv-DQN on four symbolic Pacman environemnts (Figure 2). The Graph DQN converges significantly faster to a performing control policy than the convolution-based DQN on all four environments.

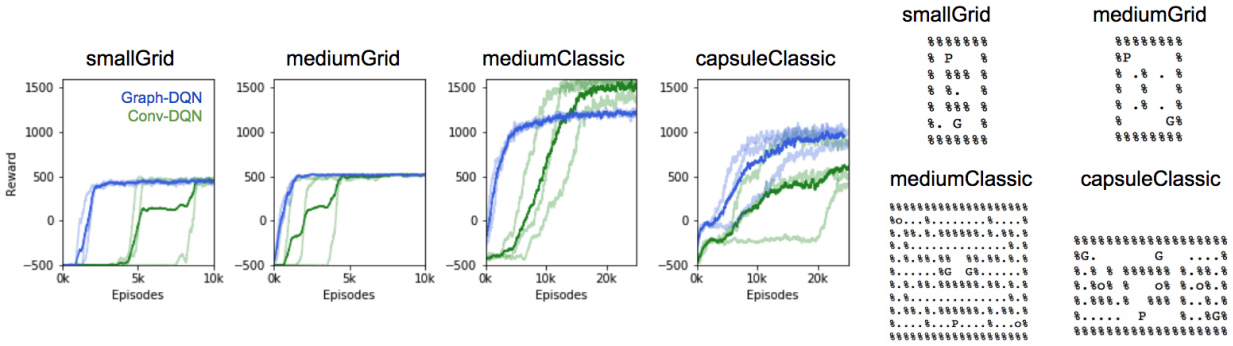


Figure 2: Pacman results. Performance of the baseline Conv-DQN (green) and GraphDQN (blue) agent on several symbolic Pacman environments (smallGrid, mediumGrid, mediumClassic, and capsuleClassic). Bold lines are the mean, with the individual $n = 3$ repetitions indicated by the lighter colors. The symbols are: `%` - wall, `P` - player, `G` - ghost, `H` - scared ghost, `.` - coin, ○ - capsule.

3.3 What do the agents learn?

To understand how the agents are interpreting the edge relations between objects, we observed the behavior of a trained agent running in an environment while manipulating the knowledge graph (Figure 3). For simplicity consider the one-one environment, with one bucket pair ($b \rightarrow B$) during training and one pair ($c \rightarrow B$) during testing. A successful behavior is shown in Figure 3a. When we removed $b \rightarrow B$, the agent still pushes the ball, but does not know where to push the ball towards, suggesting that the agent has learned to ground the feature $e_{bB} = 2$ as 'goal' or 'fills'. We swapped the edge features of $A \rightarrow B$ and $A \rightarrow b$, and the agent attempts to push the bucket into the ball. The knowledge graph could also be manipulated such the agent pushes a ball into another ball. These studies show that the agent learned the 'push' and 'fills' relation and is able to apply these actions to objects it has never pushed before.

Table 2: Manipulating Pacman behavior. Behavior and score of the Graph-DQN agent on the mediumClassic map when various edges are removed or features substituted. Reward is shown as mean± standard error over $n = 100$ repetitions.

Variation	Reward	Behavior
Base	1169 ± 39	Default behavior
Remove `Ghost→Player` edge	-170 ± 27	No clear interpretation
Set `Ghost→Player` to `Player→Coin` feature	-78 ± 38	Does not avoid ghosts
Remove `Player→Scared Ghost` edge	558 ± 25	Does not chase scared ghosts
Remove `Ghost→Scared Ghost` edge	1161 ± 29	No effect
Remove `Player→Coin` edge	-376 ± 20	Pacman moves randomly
Remove `Player→Capsule` edge	323 ± 37	Does not eat the capsule
Remove `Player→Wall` edge	-339 ± 21	Runs into the nearest wall
Remove `Ghost→Wall` edge	267 ± 33	No clear interpretation
Remove `Scared Ghost→Wall` edge	530 ± 28	Does not chase scared ghosts

Similarly, in Pacman, if we remove the `Player→Scared Ghost` edge, the agent no longer chases the scared ghosts (Table 2). Without an edge to the capsule, the agent no longer eats the capsule. The agent can also be manipulated to not avoid ghosts by changing the `Ghost→Player` feature to the `Player→Coin` edge relation.

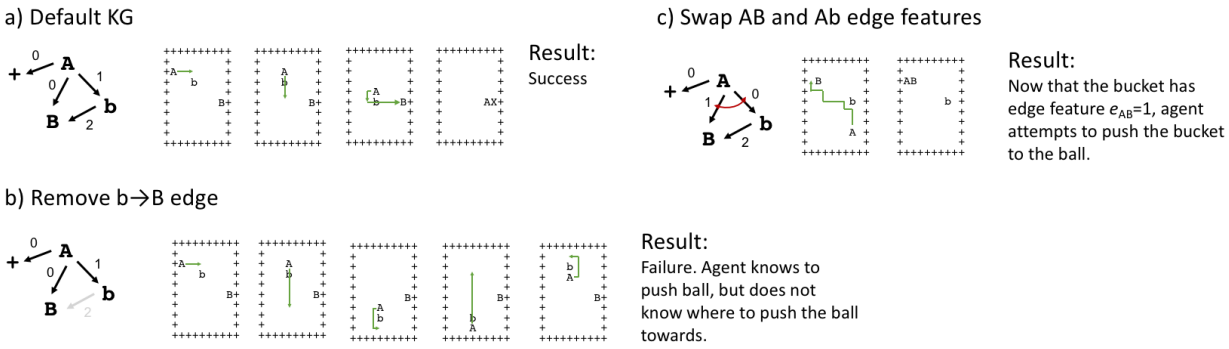


Figure 3: Manipulating Trained Agents in Warehouse. We used agents trained on the base knowledge graph (a), and manipulated their behavior at runtime by changing the input knowledge graph.

Conclusion: The field has long debated the importance of reasoning with symbols (that may incorporate prior knowledge) and its compatibility with gradient based learning. The Graph-DQN architecture provides one framework to bridge these seemingly disparate approaches [4].

References

[1] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

[2] David Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, arXiv:1712.01815:1140–1144, 2018.

[3] Volodymyr Minh et al. Human-level control through deep reinforcement learning. *Nature*, arXiv:1312.5602(7540).

[4] Marta Garnelo and Murray Shanahan. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17–23, Oct 2019.

[5] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, Nov 2016.

[6] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krahenbuhl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. arXiv:1810.12282, 2018.

[7] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017.

[8] Thomas Stepleton. The pycolab game engine, 2017.

[9] Vinicius Zambaldi. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2019.

Model-free and model-based learning processes in the updating of explicit and implicit evaluations

Benedek Kurdi
Department of Psychology
Harvard University
kurdi@g.harvard.edu

Samuel J. Gershman
Department of Psychology
Harvard University
gershman@fas.harvard.edu

Mahzarin R. Banaji
Department of Psychology
Harvard University
mahzarin_banaji@harvard.edu

Abstract

Evaluating stimuli along a positive–negative dimension is a fundamental computation performed by the human mind. In recent decades, research has documented both dissociations and associations between explicit (self-reported) and implicit (indirectly measured) forms of evaluations. Together, these two forms of evaluation are central to organizing social cognition and drive behavior in intergroup relations, consumer choice, psychopathology, and close relationships. However, it is unclear whether explicit–implicit dissociations arise from relatively more superficial differences in measurement techniques or from deeper differences in the processes by which explicit and implicit evaluations are acquired and represented. The current project (total sample size: $N = 2{,}354$) relies on the computationally well-specified distinction between model-based and model-free reinforcement learning to investigate the unique and shared aspects of explicit and implicit evaluations. Study 1 used a revaluation procedure to reveal that whereas explicit evaluations of novel targets are updated via both model-free and model-based processes, implicit evaluations depend on the former but are impervious to the latter. Studies 2–3 demonstrated the robustness of this effect to (*a*) the number of stimulus exposures in the revaluation phase and (*b*) the deterministic vs. probabilistic nature of initial reinforcement. These findings provide a novel framework, going beyond traditional dual-process and single-process accounts, to highlight the context-sensitivity and long-term recalcitrance of implicit evaluations as well as variations in their relationship with their explicit counterparts. These results also suggest novel avenues for designing theoretically guided interventions to produce change in implicit evaluations.

Keywords: human cognition, Implicit Association Test, implicit evaluations, model-free vs. model-based learning, social cognition

Acknowledgements

This work was supported by the Dean's Competitive Fund for Promising Scholarship of the Harvard Faculty of Arts and Sciences and an Undergraduate Research Scholar award from the Institute of Quantitative Social Science at Harvard University (both to Mahzarin R. Banaji).

1 Background

Evaluations of entities along a positive–negative continuum (e.g., "I prefer Federer to Nadal"), also referred to as *attitudes*, are central to structuring affect, behavior, and cognition in humans. After decades of work relying on self-report measures to index such evaluations, social cognition research since the 1980s has been guided by the recognition that evaluations can also be activated automatically upon encountering a stimulus (1). These automatically activated evaluative representations (*implicit attitudes*) are measured using response interference tasks (e.g., 2). Specifically, on the Implicit Association Test (IAT) used in the present studies, implicit evaluations of two targets (e.g., Federer vs. Nadal) are inferred from the speed and accuracy of sorting stimuli representing each category together with positive words (e.g., "love" and "sunshine") vs. negative words (e.g., "hate" and "vomit"). By contrast, *explicit attitudes* are measured by self-report.

Dominant dual-process theories posit that, beyond differences in measurement, explicit and implicit evaluations also differ from each other in more profound ways. Crucially, explicit and implicit evaluations are hypothesized to originate from fundamentally different learning processes. Specifically, the learning processes giving rise to explicit evaluations are assumed to be flexible and rule-governed and to rely on propositional information, whereas the learning processes giving rise to implicit evaluations are assumed to be slow and gradual and to rely on associative regularities encountered in the environment (3).

Although this dual-process perspective on evaluative learning has inspired much work on the acquisition and change of explicit and implicit evaluations, it suffers from some notable shortcomings. First, in opposition to the theory, it has been repeatedly demonstrated that implicit evaluations can be flexibly updated via purely verbal instructions (4). Such findings have prompted some to abandon a dual-process perspective on evaluative learning and to replace it with a model of evaluative learning that relies on a single propositional process (4). Second, dual-process theories of evaluation are difficult to falsify and the same applies to single-process alternatives. For instance, whether learning is quick or slow is a matter of judgment and, as such, researchers with different theoretical commitments may make widely divergent inferences from the same data. Third, implicit evaluations exhibit a host of characteristics that are not accounted for by dual-process or single-process theories. For instance, implicit evaluations have been shown to be context-specific and situationally malleable (5). Fourth, under dual-process theories, explicit and implicit evaluations emerge from different learning processes and, as such, convergence between the two is unexpected. By contrast, under single-process theories, explicit and implicit evaluations emerge from the same learning process and, as such, they should always converge. In fact, the majority of empirical data fall between these two extremes: Explicit and implicit evaluations are typically correlated with each other but are rarely redundant (6).

2 The present project

Given these shortcomings of existing dual-process and single-process theories, the present project investigated, for the first time, whether (*a*) implicit evaluations, as measured using the Implicit Association Test (IAT) (2) are responsive to reinforcement learning, and (*b*) implicit and explicit evaluations are differentially responsive to model-free versus model-based processes (7). As such, our aim is to provide a new and computationally precise general framework in which the acquisition and change of implicit and explicit evaluations can be understood.

Although model-free and model-based algorithms solve the same reinforcement learning problem, they differ from each other both in the way they learn and the kind of information that they are able to represent (7). Model-free algorithms are experience-based and computationally cheap, and create a highly compressed representation of the past history of rewards. By contrast, model-based algorithms rely on a causal model of the environment over which mental simulations can be performed. As such, they are computationally expensive and considerably more flexible than model-free algorithms. Human learners have been shown to rely on both model-free and model-based strategies (8). Under some conditions, model-free and model-based learning can converge on the same behavioral output. However, the results of model-free and model-based learning can diverge when the environment changes in such a way as to modify the motivational relevance of a known stimulus. Specifically, a paradigm commonly referred to as *reward revaluation* has often been used to discern whether humans rely on model-free or model-based learning (8). Here we used this paradigm for the first time to probe the updating of implicit (i.e., indirectly revealed) evaluations.

The present studies (total sample size: $N = 2{,}354$) consisted of a *learning phase* and a *test phase*. In the learning phase, participants interacted with two novel groups (Laapians vs. Niffians) and received rewards (positive

points) or punishments (negative points) as a result of their choice behavior. In the test phase, they completed measures of explicit evaluation and implicit evaluation (2) of the Laapian and Niffian targets.

Crucially, for the learning phase of the experiment, participants were assigned to one of five between-subjects conditions (see Figure 1). In the **control** (Study 1) and **baseline learning conditions** (Studies 1–3), the learning phase consisted of a single part, whereas in the **reward revaluation** (Studies 1–3), **transition revaluation** (Study 1), and **relearning conditions** (Study 1), the learning phase consisted of two parts.

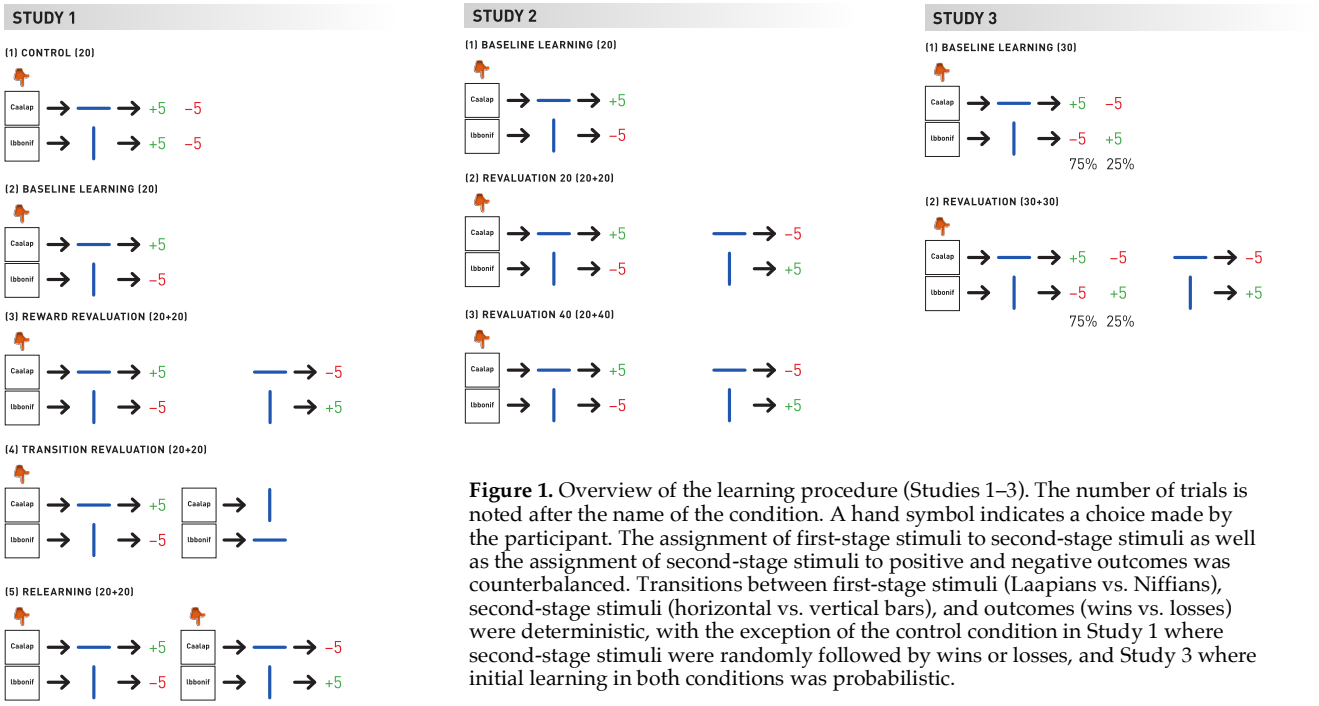


Figure 1. Overview of the learning procedure (Studies 1–3). The number of trials is noted after the name of the condition. A hand symbol indicates a choice made by the participant. The assignment of first-stage stimuli to second-stage stimuli as well as the assignment of second-stage stimuli to positive and negative outcomes was counterbalanced. Transitions between first-stage stimuli (Laapians vs. Niffians), second-stage stimuli (horizontal vs. vertical bars), and outcomes (wins vs. losses) were deterministic, with the exception of the control condition in Study 1 where second-stage stimuli were randomly followed by wins or losses, and Study 3 where initial learning in both conditions was probabilistic.

In the first part of the learning phase, participants completed learning trials on which they made a choice between a Laapian and a Niffian target. Depending on their choice, they were then exposed to a second-stage stimulus (vertical or horizontal bar), followed by a positive or negative outcome (+5 or -5 points). Participants were instructed to maximize wins. The relationship between first-stage and second-stage stimuli was deterministic (e.g., Laapians were always followed by horizontal bars and Niffians by vertical bars). In the **control condition** (Study 1), second-stage stimuli were randomly followed by wins or losses, thus providing a measure of relative preference at baseline. In **all four remaining conditions** of Studies 1–2, the transition between second-stage stimuli and rewards was deterministic, whereas in Study 3 it was probabilistic, with one second-stage stimulus followed by wins 75 percent of the time and the other second-stage stimulus followed by losses 75 percent of the time.

In the **reward revaluation**, **transition revaluation**, and **relearning conditions**, the first part of the learning phase was followed by a second part. In the **reward revaluation conditions**, the transition between second-stage stimuli and rewards was reversed compared to the first part of the learning phase (without participants making choices between first-stage stimuli). In the **transition revaluation condition**, the transition between first-stage and second-stage stimuli was reversed compared to the learning phase (without participants making any choices). The **relearning condition** was similar to the reward revaluation condition in that the transition between second-stage stimuli and rewards was reversed; however, participants experienced the full transition structure from first-stage stimuli to second-stage stimuli to rewards based on their choices between Laapian and Niffian targets.

As such, the **control condition** (Study 1) indexes explicit and implicit evaluations without any meaningful learning. The **baseline learning condition** (compared to the control condition; Study 1) probes whether explicit and implicit evaluations are updated in the face of valenced feedback involving novel stimuli. Crucially, the **reward revaluation** and **transition revaluation conditions** (compared to the baseline learning condition; Study 1) test whether explicit and implicit evaluations are (differentially) reflective of revaluation (i.e., model-based learning). Studies 2 and 3 investigate the same issue by asking whether reward revaluation shifts explicit and implicit attitudes when (*a*) the revaluation phase involves twice as many trials as the base-

line learning phase (Study 2) and (*b*) initial reinforcement is probabilistic rather than deterministic (Study 3). Finally, to the extent that implicit evaluations do not show updating in the revaluation conditions, the **relearning condition** (compared to the baseline learning condition; Study 1) can be used to establish whether implicit evaluations, once in place, do not respond to any kind of new information, or they are specifically impervious to model-based, but not model-free, updating.

3 Results

3.1 Explicit evaluations

Results using explicit measures of evaluation replicated well-established results regarding the sensitivity of such evaluations to both model-free and model-based learning (8). In Study 1, baseline learning shifted explicit evaluations compared to control, $t(548.86) = 9.88$, $P < 0.0001$, $BF_{10} = 3.40 \times 10^{18}$. Similarly, reward revaluation shifted explicit evaluations compared to the baseline learning condition, $t(474.09) = 14.49$, $P < 0.0001$, $BF_{10} = 5.89 \times 10^{38}$. The same result was observed for transition revaluation, $t(502.54) = 10.44$, $P < 0.0001$, $BF_{10} = 5.06 \times 10^{20}$. Finally, the relearning condition was also found to shift explicit evaluations, $t(793.91) = 24.55$, $P < 0.0001$, $BF_{10} = 1.62 \times 10^{85}$. In Study 2, explicit evaluations shifted significantly as a result of revaluation in both the revaluation 20 condition, $t(124.17) = 8.31$, $P < 0.0001$, $BF_{10} = 2.04 \times 10^{12}$, and in the revaluation 40 condition, $t(115.47) = 7.42$, $P < 0.0001$, $BF_{10} = 1.44 \times 10^{10}$, thus replicating the results of Study 1. In Study 3, explicit evaluations were again found to shift significantly as a result of reward revaluation, although the evidence in favor of change was weaker than in Studies 1 and 2, $t(351.96) = 2.16$, $P = 0.031$, $BF_{10} = 1.10$.

3.2 Implicit evaluations

Implicit evaluations, like explicit evaluations, were sensitive to reinforcement learning, as shown by a significant difference between the control and baseline learning conditions, $t(565.06) = 4.35$, $P < 0.0001$, $BF_{10} = 9.11 \times 10^{3}$. Unlike with explicit measures, the crucial comparison between the baseline learning and reward revaluation conditions provided evidence in favor of the null hypothesis, $t(569.05) = 1.06$, $P = 0.287$, $BF_{01} = 6.22$, suggesting that implicit evaluations are impervious to model-based updating. Moreover, we found only weak evidence that the transition revaluation condition may have differed from the baseline learning condition, $t(591.29) = 2.47$, $P = 0.013$, $BF_{10} = 1.85$. However, this small difference does not necessarily indicate the use of model-based learning: Given that it had been paired with reward, the second-stage stimulus could have served as a reinforcer (akin to second-order conditioning). Finally, given that no updating was found in the reward revaluation condition, a comparison involving the baseline learning and relearning conditions can be used to establish whether implicit evaluations are (*a*) generally impervious to updating or (*b*) more specifically impervious to model-based, but not model-free, updating. The baseline learning and relearning conditions were found to significantly differ from each other, $t(649.58) = 6.04$, $P < 0.0001$, $BF_{10} = 2.40 \times 10^{6}$, suggesting that already established implicit evaluations can be effectively updated provided that such updating can be performed via model-free mechanisms.

Replicating the results of Study 1, implicit evaluations were found to be impervious to reward revaluation in both Study 2 and Study 3. Specifically, Study 2 provided evidence in favor of the null hypothesis when the number of trials was the same across the first and second parts of the learning phase (baseline learning vs. revaluation 20 conditions), $t(154.74) = -0.87$, $P = 0.381$, $BF_{01} = 4.19$. A similar result was obtained in the revaluation 40 condition where the number of revaluation trials was double the number of the initial learning trials, $t(152.71) = -0.51$, $P = 0.612$, $BF_{01} = 5.28$. Implicit evaluations also remained insensitive to reward revaluation in Study 3, demonstrating that their insensitivity to model-based learning does not depend on the deterministic vs. probabilistic nature of initial reinforcement, $t(366.99) = 0.32$, $P = 0.747$, $BF_{01} = 8.26$.

4 Discussion

The novel contribution of the present project is twofold. First, we have shown that implicit (automatically and indirectly revealed) evaluations of stimuli, like their explicit (self-reported) counterparts, are amenable to updating as a result of reinforcement learning, i.e., experience with the positive and negative outcomes of actions involving those stimuli. Second, we have demonstrated both a commonality and a difference in the computations underpinning the updating of explicit vs. implicit evaluations via reinforcement learning: Just like explicit evaluations, implicit evaluations were found to be responsive to model-free processes. However, unlike their explicit counterparts, implicit evaluations were insensitive to model-based learning.

This perspective readily explains the hitherto puzzling finding that whereas certain interventions can successfully shift implicit evaluations, others seem to be completely ineffective. Specifically, out of 17 interventions implemented in a recent large-scale collaboration (9), only eight shifted implicit evaluations of African Americans toward neutrality, while nine produced no change. Five of the eight effective interventions required no model of the environment, whereas three required a model of the simplest possible form [P(positive | African American) = P(negative | White American) = 1]. By contrast, all but one ineffective interventions involved a complex causal model of the environment (e.g., a model of another persons' mind, a model of a positive encounter with an outgroup member, or a model of racial injustice). The one remaining unsuccessful intervention required no model; however, it involved both rewards and punishments following both White American and African American targets and, as such, should not produce learning.

Although the present studies created a pattern of dissociation between explicit and implicit evaluations, it should be noted that a reinforcement learning perspective, unlike a traditional dual-process view (3), does not make an unqualified prediction of explicit–implicit dissociations, for multiple reasons. First, in many situations, including the baseline learning condition of the present studies, model-free and model-based algorithms converge on the same value representation. Second, as demonstrated by the present studies, explicit and implicit evaluations can both be updated by model-free processes. This shared learning process should generally lead to some degree of association between explicit and implicit evaluations. Third, recent research has shown that model-free and model-based learning need not be antagonistic: On the contrary, a model of the environment can be used to modulate model-free value representations via simulated experience (10). Future work may test this idea in the context of implicit evaluations by imposing a delay between the revaluation and test phases of the experiment.

Moreover, model-free learning is inherently state-dependent, which may account for the highly contextualized nature of implicit evaluations (5) as well as their resistance to long-term change. By mapping out the space of relevant states and providing model-free training across a large number of them, change in implicit evaluations may become more robust, enduring, and generalizable.

5 References

1. Greenwald AG, Banaji MR (1995) Implicit social cognition: Attitudes, self-esteem, and stereotypes. *Psychol Rev* 102(1):4–27.

2. Greenwald AG, McGhee DE, Schwartz JLK (1998) Measuring individual differences in implicit cognition: The Implicit Association Test. *J Pers Soc Psychol* 74(6):1464–1480.

3. Rydell RJ, McConnell AR (2006) Understanding implicit and explicit attitude change: A systems of reasoning analysis. *J Pers Soc Psychol* 91(6):995–1008.

4. De Houwer J (2014) A propositional model of implicit evaluation. *Soc Pers Psychol Comp* 8(7):342–353.

5. Blair IV (2002) The malleability of automatic stereotypes and prejudice. *Pers Soc Psychol Rev* 6(3):242–261.

6. Nosek BA (2005) Moderators of the relationship between implicit and explicit evaluation. *J Exp Psychol Gen* 134(4):565–584.

7. Sutton RS, Barto AG (1998) *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA).

8. Daw ND, Niv Y, Dayan P (2005) Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nat Neurosci* 8(12):1704–1711.

9. Lai CK, et al. (2014) Reducing implicit racial preferences: I. A comparative investigation of 17 interventions. *J Exp Psychol Gen* 143(4):1765–1785.

10. Gershman SJ, Zhou J, Kommers C (2017) Imaginative reinforcement learning: Computational principles and neural mechanisms. *J Cognitive Neurosci* 29(12):2103–2113.

A Comparison of Non-human Primate and Deep Reinforcement Learning Agent Performance in a Virtual Pursuit-Avoidance Task

Theodore L. Willke
Intel Labs
Intel Corporation
Hillsboro, OR USA
ted.willke@intel.com

Seng Bum M. Yoo
Department of Neuroscience
University of Minnesota
Minneapolis, MN USA
sbyoo.ur.bcs@gmail.com

Mihai Capotă
Intel Labs
Intel Corporation
Hillsboro, OR USA
mihai.capota@intel.com

Sebastian Musslick
Princeton Neuroscience Institute
Princeton University
Princeton, NJ USA
musslick@princeton.edu

Benjamin Y. Hayden
Department of Neuroscience
University of Minnesota
Minneapolis, MN USA
benhayden@gmail.com

Jonathan D. Cohen
Princeton Neuroscience Institute
Princeton University
Princeton, NJ USA
jdc@princeton.edu

Abstract

We compare the performance of non-human primates and deep reinforcement learning agents in a virtual pursuit-avoidance task, as part of an effort to understand the role that cognitive control plays in the deeply evolved skill of chase and escape behavior. Here we train two agents, a deep Q network and an actor-critic model, on a video game in which the player must capture a prey while avoiding a predator. A previously trained rhesus macaque performed well on this task, and in a manner that obeyed basic principles of Newtonian physics. We sought to compare the principles learned by artificial agents with those followed by the animal, as determined by the ability of one to predict the other. Our findings suggest that the agents learn primarily 1st order physics of motion, while the animal exhibited abilities consistent with the 2nd order physics of motion. We identify scenarios in which the actions taken by the animal and agents were consistent as well as ones in which they differed, including some surprising strategies exhibited by the agents. Finally, we remark on how the differences between how the agents and the macaque learn the task may affect their peak performance as well as their ability to generalize to other tasks.

Keywords: deep reinforcement learning, pursuit tasks, motion prediction

1 Introduction and background

There is mounting interest in comparing the performance of artificial agents with natural agents in complex task domains. While in some domains, artificial agents have been successful in outperforming humans (*1–3*), there are other domains in which they still fail to do so (*4*). On the one hand, the systematic comparison of artificial and natural agents can help identify weaknesses in state-of-the-art artificial intelligence (AI) and, on the other hand, it can help to better understand the cognitive processes, and underlying neural mechanisms responsible for the behavior of natural agents, including humans. However, comparisons between artificial and natural agents are often limited to crude behavioral measures, such as overall task performance, and are constrained by a lack of experimental control over the task environment.

Here, we compare the performance of two deep reinforcement learning (RL) agents and a rhesus macaque in a pursuit avoidance task that is complex enough to challenge to state-of-the art reinforcement learning models but simple enough to train non-human primates to high levels of performance. In this paradigm, the agent (used henceforth to refer to both the non-human primates and deep RL agents) is faced with continuous decisions about where to move based on the position of a predator that the agent must avoid and the position of a prey that the agent must catch to be rewarded. The prey and predator follow cost-driven movement policies that involve repulsion from and attraction to the agent, respectively. These highly interactive dynamics, paired with sparse rewards and punishments contingent on avoiding the predator and catching the prey, present a significant challenge to reinforcement learning models. Here we train a deep Q-learning network, as well as a deep actor-critic model to perform this task, and compare the two artificial agents against the non-human primates with respect to (a) overall performance on task, (b) learning dynamics, (c) behavioral strategies as a function of prey and predator position, and (d) the complexity of the internal model of the task that is reflected in the agent's decision-making behavior. We discuss implications of this analysis for the development of future artificial agents, as well as for understanding the latent cognitive variables underlying decision-making processes in non-human primates.

2 Methods

2.1 Virtual pursuit-avoidance task design

We designed the virtual pursuit-avoidance task based on an existing visual experimental design (*5*). In this design, the agent controls the position of an avatar (yellow circle in Fig. 1). The game determines the predator and prey's next positions at each step according to attraction and repulsion force functions, along with a cost contour map over the pixel field (1920 by 1080 pixels). The field cost is higher along the perimeter to mitigate cornering tactics. The color of the predator and prey objects encodes their maximum speed. The agent is always yellow. Unless otherwise stated, the starting points of the objects and their colors are randomized trial-by-trial, with a minimum starting distance to the agent of 400 pixels.

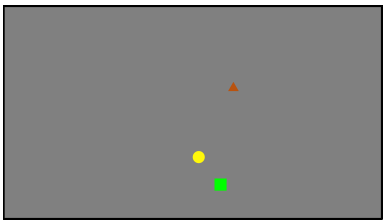


Figure 1: The visual environment. The agent (yellow circle), predator (red triangle), and prey (green square) are shown.

The macaque interacted with the environment through a joystick providing proportional inputs in two dimensions ($-1 \leq x, y \leq 1$). The deep RL agents interacted through an environment API. The API provides an `(observation, reward, done)` tuple each trial step and accepts an action vector. The observation consisted of the color values and x-y coordinates of the agent, predator, and prey. Steps were assigned a small penalty, predators a large penalty, preys a large reward, and timeouts (max steps reached) a medium penalty. The action vector was agent-dependent (see below). The original environment was implemented in Matlab using Psychtoolbox (*6*) as an interactive simulation with joystick-driven input. We implemented a new environment in Python using PsychoPy (*7*) for training the RL agents. The agent interface conforms to the de facto standard defined by the OpenAI Gym project (*8*).

2.2 Macaque training and evaluation

A male adult Macaca mulatta (8 year old) was trained using a multi-stage curriculum. The goal was to learn the following in sequence: 1) touching the joystick produces reward; 2) moving the joystick controls the position of a circle (agent's

avatar) on the screen; 3) overlap of the circle and a square (prey) on the screen provides a reward; 4) objects move; 5) color indicates the amount of reward as well as maximum speed of the prey; 6) a triangle (predator) will pursue the circle and overlap results in a penalty.

2.3 Deep reinforcement learning agents

The deep Q network implemented Q-learning, a form of off-policy temporal-difference learning that estimates an action-value function through iterative updates. This function estimates the expected value of all future rewards for a given state and action (Eqn. 1).

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') \middle| s, a \right] \tag{1}$$

Given the size of the state space, this function was approximated using a fully-connected deep neural network (*1*). The model performed online RL using two copies of the same network - the policy network and the target network. In our "double DQN" (*9*), the policy network updated the action-value function as it replayed prioritized experiences from memory. Periodically its parameters were loaded into the target network, which took action in the environment. Replay was prioritized by the magnitude of an experience's TD error (*10*). Epsilon greedy exploration was used in the bootstrapping process. A short sequence of observations were "stacked" to provide the input state to the network. The network output was a 1-hot state vector of 9 possible actions (cardinal and sub-cardinal directions and none). We used our own Python implementation.

Soft Actor-Critic (SAC) (*11*) is an off-policy actor-critic agent that simultaneously maximizes expected return and policy entropy:

$$\pi^* = \arg\max_{\pi} \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))], \tag{2}$$

SAC automatically uses approximating dual gradient descent to tune the parameter α that determines the importance of entropy. The critic part of SAC uses the minimum of two soft Q function approximators to counter positive bias when computing gradients. It also uses a target soft Q function and a replay pool, like DQN. SAC outputs continuous [-1, 1] actions, emulating joystick input. We used the Rlkit implementation of the agent (*12*).

2.4 Deep RL agent training and evaluation

The deep RL agents were trained on randomly-initialized episodes. The agents executed until one of the termination conditions - catch, caught, or maximum steps - was encountered. The agents were trained using backpropagation until catch performance stopped improving, somewhere between 6,000 and 10,000 episodes. DQN used epsilon greedy exploration, starting at 1.0 and ending at 0.25; epsilon was 0 for evaluation. SAC, which targets maximum entropy for its policy, does not inject explicit exploration noise; instead, it uses the mean action.

DQN used a frame skipping technique, processing every 3rd frame, to lower the training time without eliminating useful motion information. Memory replay capacity was 100,000 episodes, which were replayed at 128 episodes per step based on rank-based importance sampling.

SAC processed every frame and its replay memory capacity was 1,000,000 episodes, not prioritized, with a batch size of 1024. Notably, SAC achieved high performance on the task using default hyperparameters. In contrast, DQN was extremely sensitive to many of the hyperparameters listed above.

The agents were evaluated on a different set of randomly initialized episodes (>10,000). The environmental parameters used (screen size, object size, object speed, etc.) matched the parameters used with the macaque. The network model parameters were frozen during this phase (i.e., no backpropagation).

3 Results and Discussion

Qualitative comparison of training and learning characteristics: The macaque was trained on the task for 70 days and 10,850 trials, after finishing the previous curriculum stages. The agents reached peak performance in a similar number of trials. Training took 2-10 hours on single-CPU machines. The training curves are shown in Fig. 2.

Overall performance on the task: We compared overall performance after the macaque and the agents had achieved peak performance on the task. The results are shown in Table 1. Both agents greatly outperformed the macaque on the task. The superior performance of the RL agents is not attributed to a lack of attention or motivation by the macaque, a fact supported by the macaque's high performance on other types of trials (e.g., one prey, no predator) (*5*).

Evidence that the macaque and the RL agents model the physics of motion: We analyzed the step-by-step trajectories for evidence that the agents had learned to instantaneously predict prey and predator movement by modeling 1st, 2nd,

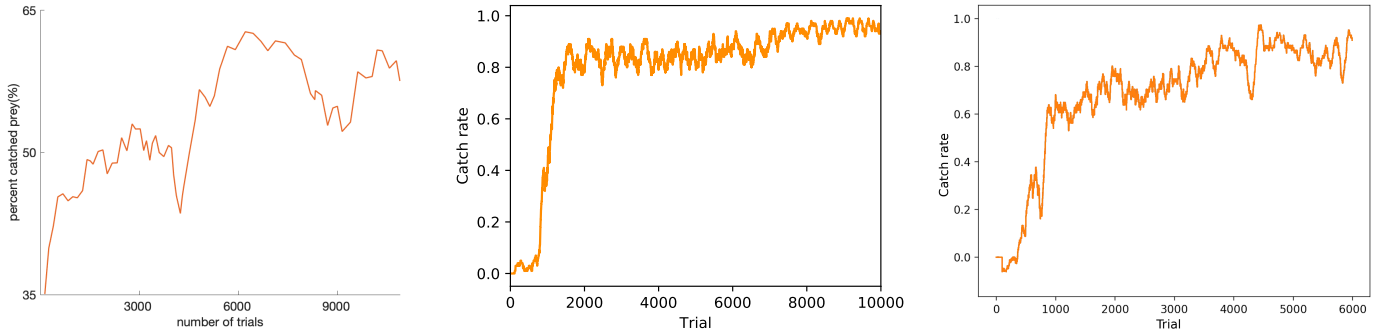


Figure 2: Training performance curves for the macaque (left), SAC agent (middle), and DQN (right). The macaque performed an average of 155 trials per day (10,850 total). The catch rate performance for the agents is averaged over a trailing sliding window of 100 episodes.

	Catch		Caught		Timeout	
Player	Percentage	Mean steps	Percentage	Mean steps	Percentage	Mean steps
MK	66.67	168	33.33	169	0.26	1200
DQN	98.5	154.8	1.29	88.5	0.21	1200
SAC	99.99	48.65	0.01	43.03	0	N/A

Table 1: Peak performance on the task for the macaque (MK), the DQN agent, and the SAC agent. Termination rates and the average number of steps to termination are shown.

and/or 3rd order equations of motion. Previous work (*5*) had found evidence that macaques learn latent representations of these motion mechanics. The generative model described in (*5*) and depicted in Fig. 3 was used. In summary, the macaque's behavior was best explained by 2nd order physics, qualitatively matching the findings of (*5*) even though the current study adds a predator to the environment. In contrast, the behavior of both artificial agents was better explained by the 1st order prediction model. One explanation for this simplified modeling is that the agents are unencumbered by mechanical inertia, in contrast to the macaque, and execute more precisely-aimed actions (on average; see below).

Evidence that the macaque and the RL agents take similar/different actions depending on context: We analyzed the actions taken by the macaque and agents under the same environmental conditions (i.e., the same avatar, predator, and prey locations, and the same values). To do this, we presented the agents with an observation that the macaque had seen and taken action on. We then measured the difference in angle of action input by the macaque and agent. We binned each measurement based on the distance of the avatar to either the predator or the prey. The results are shown in Fig. 4.

For SAC, there is strong angular agreement when the avatar is within a couple hundred pixels of the prey (prey diameter is 60). The DQN exhibits its best agreement under this circumstance as well, but the agreement is less pronounced and coherent. This may be due to observed action oscillations due to the DQN's limited one-hot action space (versus the SAC's continuous action space). A similar pattern of agreement is seen as a function of predator proximity, but the pattern is more spread out because the typical agent-predator distance spans a greater range.

Trajectory analysis. We analyzed whether the agents were more effective than the macaque when initialized from steps along trajectories in episodes that the macaque had played to termination. We conditioned the analysis on outcome: 50 episodes that terminated in prey capture and 50 that terminated in being caught by the predator. At every step, we initialized the environment and had the agents play to assess relative performance in terms of outcome and steps to termination. The results are shown in Fig. 5 (agent outcomes not shown). In general, the agents capture the prey in significantly fewer steps, but with less advantage when initialized deeper into the trajectory (closer to the prey/predator). The DQN in particular performs worse when initialized close to the prey, possibly due to action space oscillations and an inability to immediately compensate for them, given the proximity. Furthermore, agents were caught much less frequently than the macaque, even when initialized with only 10% of the trajectory remaining (i.e., in close proximity). In these cases, the number of remaining steps is higher, primarily due to them escaping and eventually pursuing the prey. Both agents survived much longer than the macaque, and often escaped to later capture the prey, when initialized close to the predator.

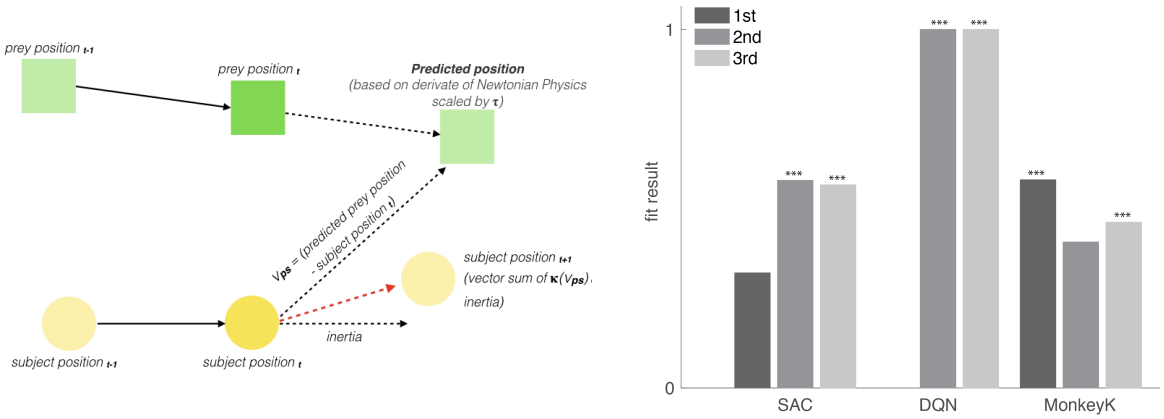


Figure 3: Fitting generative models of motion physics. The agent (yellow) moves based on past prey movement information (position, velocity, acceleration). The pursuit vector is scaled by a force parameter κ; the resulting position of the subject is then computed by summing the pursuit vector and the joystick movement inertia. Once the target location is set, an action is taken. Performance of each motion model was compared trial-by-trial and averaged across trials. Lower values indicate better fits, with significance values indicated for within-agent comparisons (stars indicate $p < 0.001$).

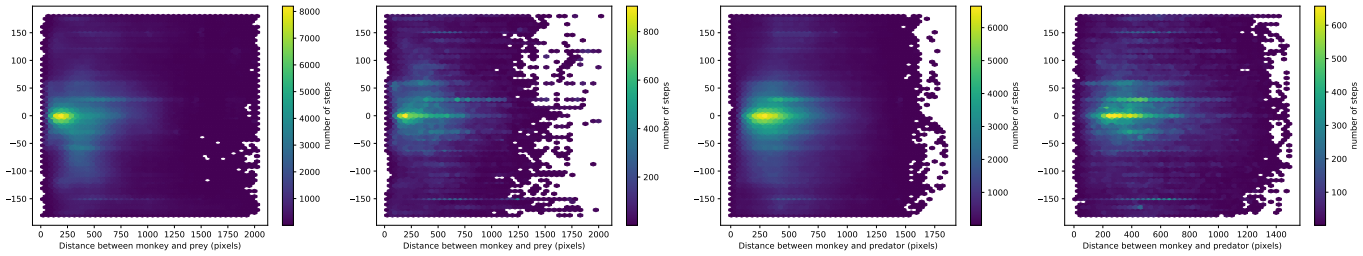


Figure 4: Action angle agreement as a function of distance to prey and predator. The color indicates the number of episode steps per bin. From the left: SAC for prey distance, DQN for prey distance, SAC for predator distance, and DQN for predator distance.

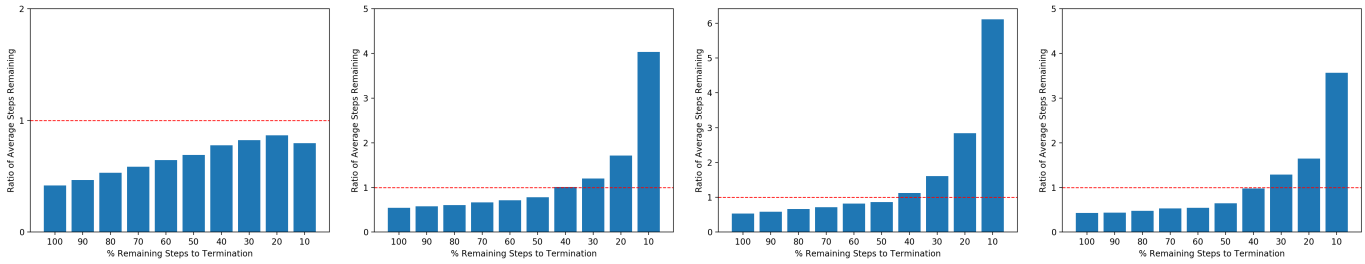


Figure 5: Trajectory analysis. From the left: SAC initialized on macaque trials that ended in prey capture; DQN for prey capture trials; SAC initialized on macaque trials that ended in being caught by predator; DQN for caught by predator trials. The RL agents were initialized progressively at steps within the trials and run to termination. Lower is better for trials that end in prey capture by agent (i.e., nearly all trials, not shown). Red line indicates that the macaque and agent would terminate in the same number of remaining steps.

4 Conclusion and Future Work

We observed that the RL agents learned as fast and to a higher level of proficiency than macaques (*5*), in contrast to reports of similar agents learning to play games much more slowly than humans. This may be due to the simplicity of our particular environment, which limits complexity to the mechanics of motion and relatively simple interactions. Peak agent performance was near perfect and they were much more efficient at achieving successful outcomes. In general, the agents exhibited a tendency to take the same action as the macaque when put into identical circumstances. However, over a series of steps the outcomes played out quite differently. The reason for this remains inconclusive. Motor control inertia and less motor precision by the macaque may explain this.

A growing number of studies suggest that humans (*13*), non-human primates (*5, 14, 15*), as well as other mammals (*16, 17*) develop predictive models of the environment to guide their decision making. In line with these results, we observed that behavior of the non-human primate can be well explained by a predictive model based on 2nd order Newtonian physics. In contrast, the behavior of trained artificial agents was best explained by a simpler model of 1st order Newtonian physics. The latter confirms recent criticisms of AI, arguing that the learning of naturalistic agents is guided by a domain-general knowledge of intuitive physics, whereas artificial agents often rely on the acquisition of domain-specific knowledge to solve the particular task with which they are confronted (*4*), without necessarily learning more general characteristics of the environment or strategies that may exploit those for other purposes. The lack of such inductive biases can result in higher performance on one task at the expense of learning efficiency and generalization performance (*18*).

The experimental setup in this work provides opportunities for a more systematic comparison of artificial and natural agents, including the opportunity to measure neural correlates of performance. The latter can be used to test for latent variables predicted by the computational models. The computational models, in turn, can be used to test the effects of constraints on processing that may help explain performance of the natural agents. For example, one candidate explanation for the lower performance of the non-human primate is a limitation in the ability to accurately perceive and/or process information about the predator and prey at the same time, thus requiring cognitive control to determine where to allocate attention (and even fixation) at a given time. Recent work suggests that such limitations take the form of a cost that attaches to the allocation of cognitive control (*19, 20*). The paradigm and computational models described here provide a platform for testing hypotheses about the nature of such costs, and the mechanisms used to evaluate them and allocate control.

References

1. V. Mnih, *et al.*, *arXiv preprint arXiv:1312.5602* (2013).
2. D. Silver, *et al.*, *nature* **529**, 484 (2016).
3. M. Campbell, A. J. Hoane Jr, F.-h. Hsu, *Artificial intelligence* **134**, 57 (2002).
4. B. M. Lake, T. D. Ullman, J. B. Tenenbaum, S. J. Gershman, *Behavioral and Brain Sciences* **40** (2017).
5. S. B. M. Yoo, S. T. Piantadosi, B. Y. Hayden, *bioRxiv* p. 272260 (2018).
6. D. H. Brainard, *Spatial Vision* **10** (1997).
7. J. W. Peirce, *Journal of Neuroscience Methods* **162**, 8 (2007).
8. G. Brockman, *et al.*, Openai gym (2016).
9. H. van Hasselt, A. Guez, D. Silver, *CoRR* **abs/1509.06461** (2015).
10. T. Schaul, J. Quan, I. Antonoglou, D. Silver, *CoRR* **abs/1511.05952** (2015).
11. T. Haarnoja, *et al.*, *CoRR* **abs/1812.05905** (2018).
12. V. Pong, https://github.com/vitchyr/rlkit.
13. N. D. Daw, Y. Niv, P. Dayan, *Nature neuroscience* **8**, 1704 (2005).
14. B. Lau, P. W. Glimcher, *Journal of the experimental analysis of behavior* **84**, 555 (2005).
15. D. Lee, M. L. Conroy, B. P. McGreevy, D. J. Barraclough, *Cognitive Brain Research* **22**, 45 (2004).
16. K. J. Miller, M. M. Botvinick, C. D. Brody, *Nature neuroscience* **20**, 1269 (2017).
17. N. Huh, S. Jo, H. Kim, J. H. Sul, M. W. Jung, *Learning & Memory* **16**, 315 (2009).
18. J. Baxter, *Learning internal representations* (Flinders University of S. Aust., 1995).
19. A. Shenhav, M. M. Botvinick, J. D. Cohen, *Neuron* **79**, 217 (2013).
20. A. Shenhav, *et al.*, *Annual review of neuroscience* **40**, 99 (2017).

Multi-batch Reinforcement Learning

Romain Laroche
Microsoft Research Montréal
romain.laroche@microsoft.com

Rémi Tachet des Combes
Microsoft Research Montréal
remi.tachet@microsoft.com

Abstract

We consider the problem of Reinforcement Learning (RL) in a multi-batch setting, also sometimes called growing-batch setting. It consists in successive rounds: at each round, a batch of data is collected with a fixed policy, then the policy may be updated for the next round. In comparison with the more classical online setting, one cannot afford to train and use a bad policy and therefore exploration must be carefully controlled. This is even more dramatic when the batch size is indexed on the past policies performance. In comparison with the mono-batch setting, also called offline setting, one should not be too conservative and keep some form of exploration because it may compromise the asymptotic convergence to an optimal policy.

In this article, we investigate the desired properties of RL algorithms in the multi-batch setting. Under some minimal assumptions, we show that the population of subjects either depletes or grows geometrically over time. This allows us to characterize conditions under which a safe policy update is preferred, and those conditions may be assessed in-between batches. We conclude the paper by advocating the benefits of using a portfolio of policies, to better control the desired amount of risk.

Keywords: Multi-Batch Reinforcement Learning, Algorithm Selection

1 Introduction

The most common setting for Reinforcement Learning (RL) is *online*: the algorithm directly interacts with the true environment and is allowed to be updated anytime. This setting is the less restrictive one from the algorithmic point of view, but real world problems (RWP) have all sorts of additional constraints that make it – most of the time – inapplicable. First example, RWP generally have a high complexity and a complete policy update would be too expensive to compute at every time step, hence the use of *online RL algorithms* that only perform small updates on the policy or the value-function estimators through temporal difference or gradient descent. The online RL algorithms comply with the complexity constraint but are less sample efficient. Second example, RWP are also generally meant to be widely deployed, on different devices with limited bandwidth, memory and computational power, which prevents frequent policy updates. As a consequence, while the online setting does not suffer from bad intermediate policies, since those can be fixed promptly, we argue that bad intermediate policies would jeopardize most RWP services.

At the opposite, the *single batch* setting, in the words of [3], refers to a reinforcement learning setting, where the complete amount of learning experience, usually a set of transitions sampled from the system, is fixed, without any access to the true environment. The literature on single batch RL focuses on safe policy improvement of the baseline policy that was used to generate the batch [6]. RWP never amount to a single policy update. Instead, we argue that most of them consist in a *multi-batch* setting, also sometimes referred to as growing batch in the literature, where the policy is successively trained on the past batches of data. This setting is commonly encountered in the following domains: dialogue systems, crop management or pharmaceutical treatment. The single batch setting might therefore be regarded as a greedily myopic study of the multi-batch setting where the former objective is a mix of safety and expected performance, neglecting the longer-term impact of the chosen policy on the quality of the next batches. As a consequence, algorithm safety might be counterproductive as it punishes exploratory strategies, which is detrimental to the asymptotic performance.

Our contributions are the following:

- We make the first attempt to model the multi-batch setting process.
- Under a set of minimal assumptions, we prove that, asymptotically, either the pool of subjects depletes, or grows geometrically.
- We conclude the paper with a set of recommendation for situational desired properties of the algorithms and argue that the situation may be assessed during the process with mild assumptions.

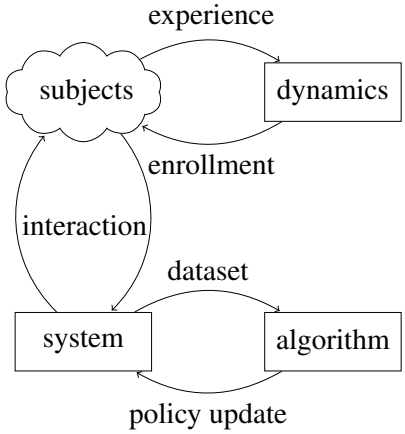


Figure 1: Multi-batch setting.

2 Multi-Batch Reinforcement Learning Process

Process 1 formalizes the generic process involved in the multi-batch setting: at every batch, the RL algorithm trains/updates a policy (Step 1). This policy is used to collect a dataset (Step 3) through interactions with a set of subjects, whose enrollment depends on the past subjects experience (Step 2). Figure 1 is an illustration of the multi-batch setting.

Step 2 is generally overlooked in the literature. However, we will show that it is crucial. Indeed, the size of the dataset $\mathcal{D}_\beta$, called crowd and denoted by $\kappa_\beta = |\mathcal{D}_\beta|$ in the following, is dependent on the past subject experience. For instance, if the algorithm generates a bad policy, it is likely to lose its subjects and later, it may only get a handful of additional experience in the next batch. Then, it may be slow to regain subjects' trust.

The goal is to optimize the cumulative return after $B \in \mathbb{R}^+$ batches. More formally, we have:

$$\mathcal{J}\left(\alpha, \kappa_0, \{\pi_0, \mathcal{D}_0\}, B\right) = \sum_{\beta=1}^{B} \sum_{k=1}^{\kappa_\beta} \dot{\rho}_{\pi_\beta, \tau_k} = \mathcal{J}\left(\alpha, \kappa_1, \{\pi_0, \mathcal{D}_0\} \cup \{\pi_1, \mathcal{D}_1\}, B-1\right) + \sum_{\tau=1}^{\kappa_1} \dot{\rho}_{\pi_1, \tau}, \tag{1}$$

Process 1: Multi-batch setting process

Input: Initial policy π_0 **Input:** Initial crowd κ_0 **Input:** Unknown environment MDP: $M = \langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$

Input: Initial dataset $\mathcal{D}_0$ **Input:** Multi-batch algorithm α **Input:** Horizon of the process (number of batches): B

for *each batch* $\beta \in [\![1, B]\!]$ **do**

Step 1: with α, train the new policy π_β on past datasets and their behavioural policies: $\pi_\beta \sim \alpha\left(\{\pi_{\beta'}, \mathcal{D}_{\beta'}\}_{\beta' \in [\![0, \beta-1]\!]}\right)$.

Step 2: enroll a crowd of κ_β subjects, in function of the past subjects experience: $\kappa_\beta \sim g\left(\{\mathcal{D}_{\beta'}\}_{\beta' \in [\![0, \beta-1]\!]}\right)$.

Step 3: collect dataset $\mathcal{D}_\beta$ of size κ_β, by following policy π_β: $\mathcal{D}_\beta = \left\{\tau_k \sim \langle \mathcal{X}, \mathcal{A}, P, \pi_\beta, R, \gamma \rangle\right\}_{k \in [\![1, \kappa_\beta]\!]}$.

end for

where k is the index of trajectory τ_k, β is a batch index, α is a multi-batch RL algorithm, π_β is the policy trained by algorithm α at batch β on dataset $\bigcup_{\beta'=0}^{\beta-1}\{\pi_{\beta'}, \mathcal{D}_{\beta'}\}$, $\dot{\rho}_{\pi_\beta,\tau_k}$ is the random variable denoting the performance of trajectory τ_k, when following policy π_β, κ_β is the crowd at batch β, π_0 is the initial policy, and $\mathcal{D}_0$ denotes the possibly empty initial batch of data.

In the following, the performance $\dot{\rho}_{\pi_\beta,\tau}$ of a given trajectory τ will be defined as the classical infinite horizon RL discounted return: $\dot{\rho}_{\pi_\beta,\tau} = \sum_{t=0}^{\infty}\gamma^t r_{\pi_\beta,\tau,t}$, (we assume that all trajectories of batch $B-1$ have terminated before starting batch B) but any other trajectory-sighted objective function may be considered and, at the exception of Proposition 2 (for which similar bounds may still be found under some other mild assumptions), all results hold. For instance, $\dot{\rho}_{\pi_\beta,\tau}$ may be defined as the binary task completion.

Remark 1. *We make the following remarks about Process 1:*

(i) Step 1: some algorithms are randomized, hence the sampling sign '$\sim$'.

(ii) Step 1: algorithms cannot be considered monotonous with respect to the samples they are trained on. Indeed, some data may be misleading and lead to bad policies [4]. Some algorithms (such as vanilla model-based RL) may train policies that are performing worse with larger datasets, even in expectation [5].

(iii) Step 2: the function g for the crowd update is stochastic, hence the sampling sign '$\sim$'.

(iv) Step 2: the function g depends on individual factors: "did the subject have a good past experience with the system?" ; and global factors: "does the system have a good image?", "what is the pool size for the crowd?".

(v) Step 2: the function g is dependent on the task. In some domains, it may be unacceptable for the system to fail: it is essentially evaluated on its efficiency (e.g. autonomous cars). In others, it is acceptable for it to fail regularly (e.g. dialogue systems).

(vi) Step 2: the function g may not be monotonous: it has happened in the past that some systems got hyped because they were failing in an entertaining way. We may cite three famous examples: Tay, Baidu and Youtube Rewind 2018.

(vii) Step 3: the dataset collection involves several sources of stochasticity: π_β, P, and R.

(viii) Steps 2 & 3: subjects behave differently from one another and overtime.

3 Analysis

In order to give some insights on the dynamics, and similarly to [2], we make a series of assumptions that should account for a large variety of multi-batch RL settings:

Assumption 1. *The multi-batch RL process is simplified as follows:*

(i) The performance $\dot{\rho}_{\pi_\beta,\tau}$ of trajectory τ generated at batch β is a random variable that belongs to $[-1,1]$.

(ii) The crowd κ_β at any given batch β is assumed to be subject-centered, i.i.d., linearly bounded, and stationary over time.

Assumption 1*(ii)* states that each subject having followed a trajectory τ during batch β enrolls $\dot{g}_{\pi_\beta,\tau} \in \mathbb{N}$ subjects for the next batch. $\dot{g}_{\pi_\beta,\tau}$ is assumed to be a random variable that only depends on its last individual experience and that is bounded by some maximal value $\dot{g}_{max}$. While $\dot{\rho}_{\pi_\beta,\tau}$ and $\dot{g}_{\pi_\beta,\tau}$ are only depending on the generated trajectory, it is more convenient to consider them as being directly sampled from a distribution only dependent on the policy π_β: $\dot{\rho}_{\pi_\beta,\tau} \sim \dot{\rho}(\pi_\beta)$ and $\dot{g}_{\pi_\beta,\tau} \sim \dot{g}(\pi_\beta)$, but one has to keep in mind that $\dot{\rho}_{\pi_\beta,\tau}$ and $\dot{g}_{\pi_\beta,\tau}$ are thus correlated through τ.

Assumption 2. *Additionally, we assume that the random function $\dot{g}(\pi)$ is Λ-Lipschitz with respect to π and $\ell_{1,\infty}$:*

$$\mathbb{E}_\tau\left[|\dot{g}_{\pi_1,\tau} - \dot{g}_{\pi_2,\tau}|\right] \leq \Lambda\|\pi_1 - \pi_2\|_{1,\infty}, \tag{2}$$

where $\ell_{1,\infty}$ is defined as follows: $\|\pi_1 - \pi_2\|_{1,\infty} = \sup_{x\in\mathcal{X}}\int_{a\in\mathcal{A}}|\pi_1(a|x) - \pi_2(a|x)|da$.

Assumption 2 is satisfied in most cases, and in particular when the trajectory length is bounded by t_{max}. We denote the empirical mean performance during batch β with the random variable $\widehat{\rho}_\beta$ and the empirical batch growth with $\widehat{g}_\beta$:

$$\widehat{\rho}_\beta = \frac{1}{\kappa_\beta}\sum_{\tau=1}^{\kappa_\beta}\dot{\rho}_{\pi_\beta,\tau} \qquad \text{and} \qquad \widehat{g}_\beta = \frac{1}{\kappa_\beta}\sum_{\tau=1}^{\kappa_\beta}\dot{g}_{\pi_\beta,\tau}. \tag{3}$$

Then, we may write:

$$\kappa_\beta = \kappa_{\beta-1}\widehat{g}_{\beta-1} = \kappa_0\prod_{\beta'=0}^{\beta-1}\widehat{g}_{\beta'}, \tag{4}$$

As a consequence of the assumption on κ_β's dynamics, if $\dot{g}_{\pi,\tau}$ can take values greater than 1 for some policy π, then the crowd is automatically assumed unbounded.

This section makes the analysis of the multi-batch process under Assumptions 1 and 2 in a tacit manner. Due to space constraints, all the proofs are omitted in the extended abstract.

Proposition 1. *The objective function $\mathcal{J}$ unfolds as follows:*

$$\mathcal{J}\left(\alpha, \kappa_0, \{\pi_0, \mathcal{D}_0\}, B\right) = \kappa_0 \sum_{\beta=1}^{B} \widehat{\rho}_\beta \prod_{\beta'=0}^{\beta-1} \widehat{g}_{\beta'}. \tag{5}$$

From now on, we focus the analysis on the asymptotic behaviour of $\mathcal{J}$ with respect to B.

Assumption 3. *We assume that π_β uniformly converges in probability with respect to the $\ell_{1,\infty}$-norm to some policy π_∞ as β tends to infinity:*

$$\forall \epsilon > 0, \exists \beta_0 \in \mathbb{N}, \text{ such that }, \forall \beta > \beta_0, \|\pi_\beta - \pi_\infty\|_{1,\infty} < \epsilon. \tag{6}$$

Assumption 3 only states the convergence of π_β, it does not say anything on the quality of the policy π_∞ in the limit. This assumption generally holds for unbiased algorithms since the incoming data depletes either in quantity: the crowd equals 0 at some batch and thereafter, no more data may be collected, or in informative content as a consequence of the strong law of large numbers, and of the increasing nature of the data collection (as long as the trained policies do not oscillate between several equally (sub-)optimal solutions). More specifically, in finite spaces $\mathcal{X}$ and $\mathcal{A}$, this assumption is satisfied by most greedy-in-the-limit algorithms. In continuous space, it has to be noted that most algorithms include a bias that makes them sensitive to the data distribution, and this distribution is in turn dependent on the previously used policy. In case of crowd depletion, there is also the possibility for the algorithm to be randomized and therefore to generate a different policy at every batch. But, since these policies are never used because of the crowd depletion, this special case could be treated separately in a trivial way, and would not be a real issue for the generality of the theory.

Proposition 2. *We consider $t \in \mathbb{N}$ and two policies π_1 and π_2. Then, with probability larger than $1 - t\|\pi_1 - \pi_2\|_{1,\infty}$:*

$$|\dot{\rho}_{\pi_1,\tau} - \dot{\rho}_{\pi_2,\tau}| \leq 2\gamma^t.^{1} \tag{7}$$

Corollary 1. *If $\|\pi_1 - \pi_2\|_{1,\infty} < \epsilon$, then, with probability larger than $1 + \epsilon\left\lfloor \log_{\gamma^{-1}} \epsilon \right\rfloor$: $|\dot{\rho}_{\pi_1,\tau} - \dot{\rho}_{\pi_2,\tau}| \leq 2\epsilon$.*

Proposition 2 and its corollary relate the distance between the trajectories generated by two policies with their distance.

Proposition 3. *If $\|\pi_1 - \pi_2\|_{1,\infty} < \epsilon$, then, $\dot{g}_{\pi_1,\tau}$ and $\dot{g}_{\pi_2,\tau}$ are close in mean and variance:*

$$|\mathbb{E}\dot{g}_{\pi_1,\tau} - \mathbb{E}\dot{g}_{\pi_2,\tau}| \leq \Lambda\epsilon \qquad \text{and} \qquad |\mathbb{V}\dot{g}_{\pi_1,\tau} - \mathbb{V}\dot{g}_{\pi_2,\tau}| \leq \Lambda\epsilon\dot{g}_{max}, \tag{8}$$

and are equal with probability higher than $1 - \Lambda\epsilon$.

Proposition 3 relates the distance between the enrollments implied by two policies with their distance.

Corollary 2. *Under Assumption 3, $\dot{\rho}_{\pi_\beta,\tau}$ and $\dot{g}_{\pi_\beta,\tau}$ respectively converge in probability to $\dot{\rho}_{\pi_\infty,\tau} \sim \dot{\rho}(\pi_\infty)$ and $\dot{g}_{\pi_\infty,\tau} \sim \dot{g}(\pi_\infty)$:*

$$\forall \epsilon > 0, \lim_{\beta\to\infty} \mathbb{P}\left(|\dot{\rho}_{\pi_\beta,\tau} - \dot{\rho}_{\pi_\infty,\tau}| > \epsilon\right) = 0, \tag{9}$$

$$\forall \epsilon > 0, \lim_{\beta\to\infty} \mathbb{P}\left(|\dot{g}_{\pi_\beta,\tau} - \dot{g}_{\pi_\infty,\tau}| > \epsilon\right) = 0. \tag{10}$$

Corollary 2 proves, under Assumption 3, the convergence in probability of $\dot{\rho}_{\pi_\beta,\tau}$ and $\dot{g}_{\pi_\beta,\tau}$, when β tends to ∞. Now, we define two modes of asymptotic behaviour of the multi-batch process. Then, we show as our main contribution that, except for identified degenerate cases, the multi-batch process follows one of the two following modes:

Definition 1. *The crowd depletion mode ($\mathcal{CDM}$) has the following properties:*

(i) The crowd converges to 0 almost surely: $\lim_{\beta\to\infty} \kappa_\beta = 0$.

(ii) The dataset remains finite: $|\bigcup_{\beta\in\mathbb{N}} \mathcal{D}_\beta| < \infty$.

(iii) The objective function $\mathcal{J}$ remains finite: $\lim_{B\to\infty} \mathcal{J} < \infty$.

[1] The use of the same subscript τ for both random variables $\dot{\rho}_{\pi_1,\tau}$ and $\dot{\rho}_{\pi_2,\tau}$ indicates that as long as both policies behave the same, the other random events follow the same realization. In other words, they behave as generated with the same random seed.

Definition 2. *The geometric crowd mode ($\mathcal{GCM}$) has the following properties:*

(i) The crowd asymptotically grows geometrically with ratio $\mathbb{E}\dot{g}_{\pi_\infty,\tau}$ as a function of β.

(ii) The dataset size grows to infinity: $|\bigcup_{\beta\in\mathbb{N}} \mathcal{D}_\beta| = \infty$.

(iii) The objective function $\mathcal{J}$ asymptotically grows geometrically with ratio $\mathbb{E}\dot{g}_{\pi_\infty,\tau}$ as a function of B. As a consequence, $\mathcal{J}$ diverges either to $+\infty$ or $-\infty$, according to the sign of $\mathbb{E}\dot{\rho}_{\pi_\infty,\tau}$. If $\mathbb{E}\dot{\rho}_{\pi_\infty,\tau} = 0$, then nothing can be said about $\lim_{B\to\infty} \mathcal{J}$.

Theorem 1. *In the degenerate case $\dot{g}_{\pi_\infty,\tau} = 1$, nothing may be said, except that it does not follow $\mathcal{GCM}$.*

If $\mathbb{E}\dot{g}_{\pi_\infty,\tau} \leq 1$, then the process almost surely enters $\mathcal{CDM}$.

If $\mathbb{E}\dot{g}_{\pi_\infty,\tau} > 1$, then with some probability $p_\infty > 0$, the process asymptotically enters $\mathcal{GCM}$. With the complementary probability $1-p_\infty$, it enters $\mathcal{CDM}$. p_∞ may be lower bounded from batch β_0 on, if β_0 is such that for all $\beta \geq \beta_0$, expectation $\mathbb{E}\dot{g}_{\pi_\beta,\tau} \geq \mu_0 > 1$, and variance $\mathbb{V}\dot{g}_{\pi_\beta,\tau} \leq \sigma_0^2$:

$$p_\infty \geq 1 - \min_{\mu\in(1,\mu_0)} \left\{ \frac{\sigma_0^2 \mu}{\kappa_{\beta_0}(\mu_0-\mu)^2(\mu-1)}; \frac{e^{-\frac{2\kappa_{\beta_0}(\mu_0-\mu)^2}{\dot{g}_{max}^2}}}{1-e^{-\frac{2\kappa_{\beta_0}(\mu-1)(\mu_0-\mu)^2}{\dot{g}_{max}^2}}} \right\}. \tag{11}$$

In particular $\mu = \frac{1}{4}\left(1+\sqrt{8\mu_0+1}\right)$ minimizes the first term (obtained with Chebyshev's bound), the second term (obtained with Hoeffding's bounds) may be shown to have a unique local minimum that does not admit a closed form expression. A numerical simulation (not reported here) shows that Equation 11 is not a tight bound and, more surprisingly, suggests that p_∞ is close to being constant when $\mu_0 - 1$ is small, and when $\kappa_{\beta_0}(\mu_0-1)$ and σ_0^2 are constant. The derivation of tighter bounds is left for future work.

4 Concluding recommendations

The goal is to maximize the objective function $\mathcal{J}$, but given the stochasticity of the process, one has to consider the *expected indirect utility*. The concept of *indirect utility* classically refers to a measurement of the satisfaction obtained by the decision maker as a function of its objective function. It is generally assumed to be monotonically increasing with the objective function, concave in $\mathbb{R}^+$, reflecting aversion to risk and diminishing marginal utility, and asymmetric with respect to the origin. The logarithm utility function, first proposed by Bernoulli, is still commonly used: $\Upsilon_{\log}(\mathcal{J}) = \text{sign}(\mathcal{J})\log(1+|\mathcal{J}|)$.

Under the log-utility, the following recommendations may be made. The goals for a multi-batch algorithm are the following, by decreasing order of importance: to make $\mathbb{E}\dot{\rho}_{\pi_\infty,\tau}$ positive, to maximize $p_\infty \mathbb{E}\left[\log \mathbb{E}_\tau \dot{g}_{\pi_\infty,\tau} | \mathcal{GCM}\right]$, and only marginally, maximize the asymptotic expected performance $\mathbb{E}\dot{\rho}_{\pi_\infty,\tau}$. In other words, the most important for a service is to make it profitable for its owner ($\dot{\rho}_{\pi_\infty,\tau} > 0$), then to make it viable ($p_\infty \mathbb{E}\left[\log \dot{g}_{\pi_\infty,\tau} | \kappa_\beta \to \infty\right]$), and only then, to make it effective ($\mathbb{E}\dot{\rho}_{\pi_\infty,\tau}$). In most cases, maximizing p_∞ and $\mathbb{E}\left[\log \dot{g}_{\pi_\infty,\tau}\right]$ is achieved through maximizing $\mathbb{E}\dot{\rho}_{\pi_\infty,\tau}$. But in some cases, a "too good" service has for consequence to lose potential future customers. This is, for instance, why planned obsolescence exists. Consequently, the multi-batch setting is multi-objective and its most salient objective is not the optimization of the performance.

The random variable $\dot{g}_{\pi,\tau}$ is generally strongly dependent on the trajectory performance random variable. Assuming past samples of $\dot{g}_{\pi,\tau}$ are observable, after several batches, one might have a good estimate of it as a random function of $\dot{\rho}_{\pi,\tau}$. Based on such a model, one is now able to assess whether risk should be taken in order to increase $\mathbb{E}\dot{g}_{\pi_\beta,\tau}$ or to be safe in order to optimize p_∞. A/B testing over two or more policies, including a safe past policy, even without online optimization, is beneficial in order to balance the desired amount of risk and explore new policies at the same time. With online optimization, an algorithm selection for RL has been shown to outperform the most efficient one in the portfolio with a minimal amount of embedded computation power [4] and bandwidth [1].

References

[1] Raphaël Féraud, Réda Alami, and Romain Laroche. Decentralized exploration in multi-armed bandits. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[2] Tatsunori B Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. *arXiv preprint arXiv:1806.08010*, 2018.

[3] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*. Springer, 2012.

[4] Romain Laroche and Raphaël Féraud. Reinforcement learning algorithm selection. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[5] Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[6] Marek Petrik, Mohammad Ghavamzadeh, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*, 2016.

SPIBB-DQN: Safe Batch Reinforcement Learning with Function Approximation

Romain Laroche*
Microsoft Research Montréal
romain.laroche@microsoft.com

Rémi Tachet des Combes*
Microsoft Research Montréal
remi.tachet@microsoft.com

Abstract

We consider Safe Policy Improvement (SPI) in Batch Reinforcement Learning (Batch RL): from a fixed dataset and without direct access to the true environment, train a policy that is guaranteed to perform at least as well as the baseline policy used to collect the data. Our contribution is a model-free version of the SPI with Baseline Bootstrapping (SPIBB) algorithm, called SPIBB-DQN, which consists in applying the Bellman update only in state-action pairs that have been sufficiently sampled in the batch. In low-visited parts of the environment, the trained policy reproduces the baseline. We show its benefits on a navigation task and on CartPole. SPIBB-DQN is, to the best of our knowledge, the first RL algorithm relying on a neural network representation able to train efficiently and reliably from batch data, without any interaction with the environment.[1]

Keywords: Batch Reinforcement Learning, Safe Reinforcement Learning, DQN, Policy Improvement

*Equal contribution

[1]This extended abstract is an excerpt of our recent paper:
Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe Policy Improvement with Baseline Bootstrapping. *In Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

1 Introduction

Most real-world Reinforcement Learning agents [17, RL] are to be deployed simultaneously on numerous independent devices and cannot be patched quickly. In other practical applications, such as crop management or clinical tests, the outcome of a treatment can only be assessed after several years. Consequently, a bad update could be in effect for a long time, potentially hurting the user's trust and/or causing irreversible damages. Devising safe algorithms with guarantees on the policy performance is a key challenge of modern RL that needs to be tackled before any wide-scale adoption.

Batch RL is an existing approach to such offline settings and consists in training a policy on a fixed set of observations without access to the true environment [10]. It should not be mistaken with the multi-batch setting where the learner trains successive policies from small batches of interactions with the environment [5, 11]. Current Batch RL algorithms are however either unsafe or too costly computationally to be applied to real-world applications.

In this paper, we focus on Safe Policy Improvement (SPI). SPI consists in safely improving a baseline policy from a batch of data. We develop a model-free version of SPI with Baseline Bootstrapping [12, 16, SPIBB]. SPIBB bootstraps the trained policy with the baseline in the state-action pair transitions that were not probed enough in the dataset. Similarly to [14], it assumes access to the baseline. Such a scenario is typically encountered when a policy is trained in a simulator and then run in its real environment, for instance in Transfer RL [19]; or when a system is designed with expert knowledge and then optimized, e.g. in dialogue applications [20, 8].

In Section 2, we recall the basics of SPIBB and implement a model-free version of the algorithm, SPIBB-DQN. SPIBB-DQN relies on pseudo-counts of the state-actions pairs, which can either be handcrafted or approximated using density models [1], and allows to apply SPIBB algorithms to tasks requiring a neural network representation. In Section 3, we apply SPIBB-DQN to a continuous navigation task of our design and to CartPole. SPIBB-DQN is, to the best of our knowledge, the first RL algorithm relying on a neural network representation able to train efficiently and reliably from batch data, without any interaction with the environment [5]. Section 4 concludes the paper and suggests possible future research directions.

2 SPI with Baseline Bootstrapping

Background: An MDP is denoted by $M = \langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ is the action space, $R(x,a) \in [-R_{max}, R_{max}]$ is the bounded stochastic reward function, $P(\cdot|x,a)$ is the transition distribution, and $\gamma \in [0,1)$ is the discount factor. The true environment is modelled as an unknown finite MDP $M^* = \langle \mathcal{X}, \mathcal{A}, R^*, P^*, \gamma \rangle$. $\Pi = \{\pi : \mathcal{X} \to \Delta_{\mathcal{A}}\}$ is the set of stochastic policies, where $\Delta_{\mathcal{A}}$ denotes the set of probability distributions over $\mathcal{A}$. The state and state-action value functions are respectively denoted by $V^\pi_M(x)$ and $Q^\pi_M(x,a)$. We define the performance of a policy by its expected return, starting from the initial state x_0: $\rho(\pi, M) = V^\pi_M(x_0)$. Given a policy subset $\Pi' \subseteq \Pi$, a policy π' is said to be Π'-optimal for an MDP M when its performance is maximal in Π': $\rho(\pi', M) = \max_{\pi \in \Pi'} \rho(\pi, M)$.

In this paper, we focus on the batch RL setting where the algorithm does its best at learning a policy from a fixed set of experience. Given a dataset of transitions $\mathcal{D} = \langle x_j, a_j, r_j, x'_j \rangle_{j \in [\![1,N]\!]}$ collected by following a baseline policy π_b, we denote by $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{R}, \widehat{P}, \gamma \rangle$ the Maximum Likelihood Estimation (MLE) MDP of the environment. $\widehat{R}$ is the reward mean and $\widehat{P}$ the transition statistics observed in the dataset. Vanilla batch RL looks for the optimal policy in $\widehat{M}$. This policy may be found indifferently using dynamic programming on the explicitly modelled MDP $\widehat{M}$, Q-learning with experience replay until convergence [17], or Fitted-Q Iteration with a one-hot vector representation of the state space [6].

As we shall see below (a fact also observed in [12]), Vanilla batch RL performs very poorly on stochastic environments because it learns from all the transitions in the dataset, including the rarely sampled ones for which the uncertainty is large and which can thus be misleading. This is not too problematic in an online setting, as the policy privileging a poor action in a given state will soon perform it and immediately correct its estimates. In a batch setting however, the bad policy might be in action for a while before its next update, leading to very poor performance on an extended period of time. A solution to this issue is to act pessimistically when the uncertainty is high: this flip side of *optimism in the face of uncertainty* [18] can be achieved by penalizing actions rarely observed in the dataset as e.g. in RaMDP [14]. An alternative option consists in bootstrapping on the baseline, a technique we now present.

SPIBB methodology: SPIBB essentially reformulates the percentile criterion [4]. It consists in optimizing the policy with respect to its performance in the MDP estimate $\widehat{M}$, while guaranteeing it to be ζ-approximately at least as good as π_b in an admissible MDP set Ξ which contains the true MDP M^* with high probability. Formally, it writes as follows:

$$\max_{\pi \in \Pi_b} \rho(\pi, \widehat{M}), \text{ where } \Pi_b \subset \{\pi \in \Pi \text{ s.t. } \forall M \in \Xi, \rho(\pi, M) \geq \rho(\pi_b, M) - \zeta\}. \tag{1}$$

Clearly, the larger the Π_b the greater the potential policy improvement and the harder the optimization problem. By appropriately balancing those opposite effects, SPIBB makes searching for an efficient and provably-safe policy tractable in terms of computer time, while allowing for potentially substantial policy improvements. To construct Π_b, we let $N_{\mathcal{D}}(x,a)$ denote the count of state-action pair (x,a) in $\mathcal{D}$. We then define a *bootstrapped set* $\mathfrak{B} \subset \mathcal{X} \times \mathcal{A}$ containing all the pairs (x,a) whose counts are smaller than a fixed parameter $N_\wedge$. In other words, $\mathfrak{B}$ contains the state-action pairs (x,a) for which the uncertainty is high. Π_b then denotes the

set of policies π that verify:

$$\forall (x,a) \in \mathfrak{B}, \pi(a|x) = \pi_b(a|x). \tag{2}$$

In the uncertain pairs, for which relying on the observed data could potentially be risky, SPIBB leans on the baseline by copying its probability to take action a. In the others, SPIBB follows classic policy optimization techniques. For finite MDPs, this may be achieved in a model-based manner by explicitly computing the MDP model $\widehat{M}$, constructing the set of allowed policies Π_b and finally searching for the Π_b-optimal policy $\pi^{\odot}_{spibb}$ in $\widehat{M}$ using policy iteration over Π_b [9, 15]. [12] prove that Π_b-SPIBB converges to a Π_b-optimal policy $\pi^{\odot}_{spibb}$ in $\widehat{M}$, and that $\pi^{\odot}_{spibb}$ is a safe policy improvement over the baseline in the true MDP M^*. They also apply SPIBB to a gridworld example and show its benefits over existing algorithms. In the following, we describe an extension of their method to function approximators.

SPIBB-DQN: DQN [13] successfully applies Q-learning to complex video games that require deep neural networks. The method uses a variety of techniques but fundamentally consists in iteratively applying the Bellman operator to learn the Q-values of the environment. DQN can easily be extended to a batch setting by replacing the experience memory used in the original algorithm with the batch we are training on. Similarly, the SPIBB policy optimization described above may be achieved in a model-free manner by sampling transitions $\langle x_j, a_j, r_j, x'_j \rangle$ from the dataset and fitting the Q-function $Q^{(t+1)}(x_j, a_j)$ to the following target $y_j^{(t)}$:

$$\begin{aligned} y_j^{(t)} &= r_j + \gamma \max_{\pi \in \Pi_b} \sum_{a' \in \mathcal{A}} \pi(a'|x'_j) Q^{(t)}(x'_j, a') \\ &= r_j + \gamma \sum_{a'|(x'_j,a') \in \mathfrak{B}} \pi_b(a'|x'_j) Q^{(t)}(x'_j, a') + \gamma \left(\sum_{a'|(x'_j,a') \notin \mathfrak{B}} \pi_b(a'|x'_j) \right) \max_{(x'_j,a') \notin \mathfrak{B}} Q^{(t)}(x'_j, a'). \end{aligned}$$

The first term r_j is the immediate reward observed during the recorded transition, the second term is the return estimate of the bootstrapped actions (where the trained policy is constrained to the baseline policy), and the third term is the return estimate maximized over the non-bootstrapped actions. We call this algorithm SPIBB-DQN as it is equivalent to DQN fitted to $y_j^{(t)}$. Note that the computation of the SPIBB targets requires the computation of the bootstrapped set $\mathfrak{B}$, which relies on an estimate of the state-action counts $\widetilde{N}_{\mathcal{D}}(x,a)$, sometimes called pseudo-counts [1, 7, 3]. The safety of SPIBB-DQN increases with $N_\wedge$. For $N_\wedge = 0$, we observe that $\mathfrak{B} = \emptyset$, and SPIBB-DQN amounts to the original DQN. As $N_\wedge \to \infty$, we see that $\mathfrak{B} = \mathcal{X} \times \mathcal{A}$, thus SPIBB-DQN becomes fully conservative and simply reproduces π_b. Intermediate values of $N_\wedge$ allow a balance between these two extreme cases.

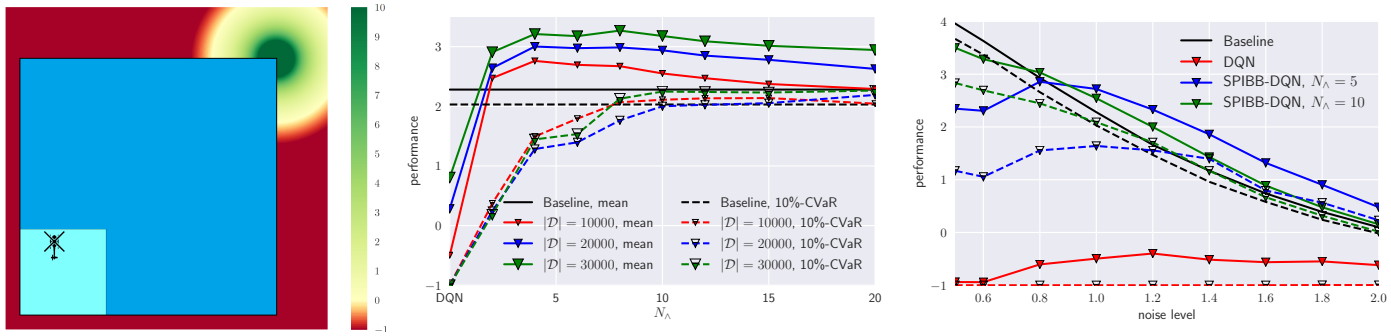
Among existing batch RL algorithms, and to the best of our knowledge, only RaMDP [14] can straightforwardly be extended to the function approximation setting. RaMDP stands for Reward-adjusted MDP and consists in modifying the observed reward by $r_j \leftarrow r_j - \kappa/\sqrt{\widetilde{N}_{\mathcal{D}}(x_j, a_j)}$ (κ is a hyperparameter of their model).

3 SPIBB-DQN empirical evaluation

The performance of Batch RL algorithms can vary greatly from one dataset to another. To properly evaluate SPIBB-DQN and its ability to produce policies that reliably outperform the baseline, we randomly generate 20 fixed-size datasets and train 15 policies on each dataset for each algorithm and for various hyper-parameter values. The algorithms are then evaluated using the mean performance and conditional value at risk performance (CVaR, also called expected shortfall) of the policies they produce. The $X\%$-CVaR is the mean performance over the $X\%$ worst runs, it is commonly used to assess the robustness of algorithms.

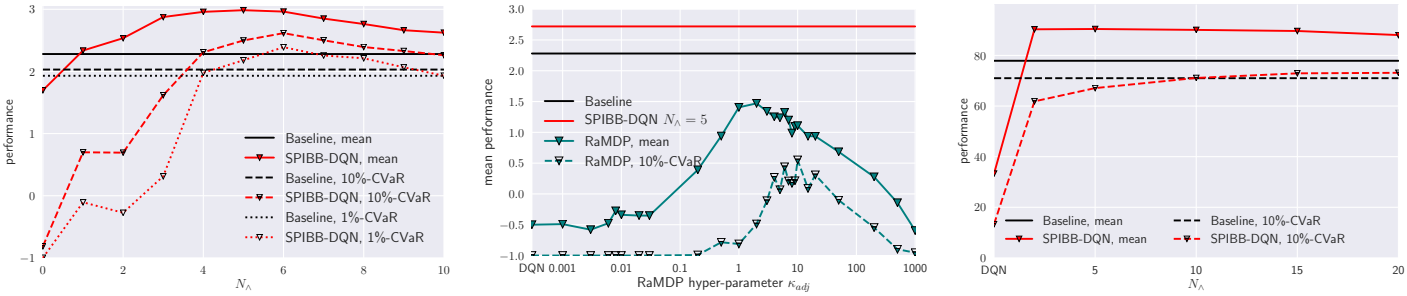
For the sake of simplicity and to be able to repeat several runs of each experiment efficiently, instead of applying costly and often finicky pseudo-count methods from the literature [1, 7, 3], we consider here a pseudo-count heuristic based on the Euclidean distance between states, and tasks where it makes sense to do so. The pseudo-count of a state-action (x,a) is defined as the sum of its similarity with the state-action pairs (x_i, a_i) found in the dataset. The similarity between (x,a) and (x_i,a_i) is equal to 0 if $a_i \neq a$, and to $\max(0, 1 - 5d(x, x_i))$ otherwise (where $d(\cdot,\cdot)$ is the Euclidean distance between two states).

We first consider a helicopter navigation task (see Figure 1(a)). The helicopter starts from a random position in the teal area, with a random initial velocity. The 9 available actions consist in applying thrust: backward, no, or forward acceleration, along the two dimensions. The episode terminates when the velocity exceeds some maximal value, in which case it gets a -1 reward, or when the helicopter leaves the blue area, in which case it gets a reward as chromatically indicated on Figure 1(a). The dynamics of the domain follow the basic laws of physics with a Gaussian centered additive noise both on the position and the velocity. To train our algorithms, we use a discount factor $\gamma = 0.9$, but report the undiscounted final reward. The baseline is generated as follows: we first train a policy with online DQN, stop before full convergence and then apply a softmax on the obtained Q-network. Our experiments consist in 300 runs on SPIBB-DQN with a range of $N_\wedge$ values and for different dataset sizes. SPIBB-DQN with $N_\wedge = 0$ is equivalent to vanilla DQN. We also perform a hyper-parameter search for RaMDP on 10k-transition datasets, with κ values ranging from 0.001 to 1000. To reduce the computational load, we only performed 75 runs per value. We also display $\kappa = 0$, which amounts to vanilla DQN. Figure 2(b) shows that, although it slightly improves over DQN, RaMDP is rather limited and stays far below the baseline.



(a) Helicopter environment. (b) Mean and 10%-CVaR vs $N_\wedge$. (c) Performance vs noise factor.

Figure 1: (a) Illustration of the environment. (b) Mean and 10%-CVaR performance as a function of $N_\wedge$ for three dataset sizes. (c) Mean and 10%-CVaR performance for the baseline, vanilla DQN, SPIBB-DQN with $N_\wedge = 5, 10$, as a function of the transition noise factor.



(a) Mean, 10%-CVaR and 1%-CVaR vs $N_\wedge$. (b) RaMDP Performance vs κ. (c) Performance on CartPole.

Figure 2: (a) Mean, 10%-CVaR and 1%-CVaR as a function of $N_\wedge$ for a single 10k dataset. (b) Mean and 10%-CVaR performance of RaMDP for multiple values of κ over the 20 10k-datasets from Figure 1. (c) Mean and 10%-CVaR performance on CartPole.

Figure 1(b) displays the mean and 10%-CVaR performances in function of $N_\wedge$ for three dataset sizes (10k, 20k, and 30k). We observe that vanilla DQN ($N_\wedge = 0$) significantly worsens the baseline in mean and achieves the worst possible 10%-CVaR performance. SPIBB-DQN not only significantly improves the baseline in mean performance for $N_\wedge \geq 1$, but also in 10%-CVaR when $N_\wedge \geq 8$. The discerning reader might wonder about the CVaR curve for the baseline. It is explained by the fact that the evaluation of the policies are not exact. The curve accounts for the evaluation errors, errors also obviously encountered with the trained policies. We performed an additional experiment. Keeping the baseline identical, we trained on 10k-transitions datasets obtained from environments with different transition noises. Figure 1(c) shows the mean and 10%-CVaR performances for the baseline, vanilla DQN, and SPIBB-DQN with $N_\wedge \in \{5, 10\}$. First, we observe that vanilla DQN performs abysmally. Second, we see that the baseline quickly gets more efficient when the noise is removed making the safe policy improvement task harder for SPIBB-DQN. As SPIBB is efficient at dealing with stochasticity, the noise attenuation reduces its usefulness. Third, as we get to higher noise factors, the stochasticity becomes too high to efficiently aim at the goal, but SPIBB-DQN still succeeds at safely improving the baseline.

Before starting the experiments reported above, we led preliminary experiments with a single 10k-transitions dataset. We found out, and report on Figure 2(a), that vanilla DQN produces very different Q-networks (and therefore very different policies) for different random seeds, even on the same dataset. It is worth noticing a posteriori that this dataset was actually favorable to DQN (mean performance of 1.7 on this dataset vs. -0.5 averaged over 20 datasets, Figure 1(b)), but that DQN's reliability is still very low. In contrast, SPIBB-DQN shows stability for $N_\wedge \geq 4$. We also report the 1%-CVaR performance (quite heavy computationally to accurately estimate) and see that there too, SPIBB-DQN performs well.

We also apply SPIBB-DQN to the CartPole environment from OpenAI gym [2]. Similarly to above, we first train a policy on the environment and build a baseline from it by applying a softmax to the learnt Q-values. To make the task more challenging once the baseline has been obtained, we add some stochasticity to the environment by acting randomly 50% of the time. With that added noise, the baseline gets an average score of approximately 78. SPIBB-DQN reaches values over 90 while remaining safe. Standard DQN gets a score of 33 and a 10%-CVaR of 13. Results can be found on Figure 2(c), for datasets of size 10k. We also applied our algorithm to Pendulum with marginal improvements (not shown here).

4 Conclusion and future work

In this paper, we study the problem of safe Batch Reinforcement Learning with function approximation. We develop a model-free version of SPIBB and demonstrate its performance on two domains. SPIBB-DQN is the first deep batch algorithm allowing policy improvement in a safe manner. Future work involves extending our results to more complex domains, *e.g.* with pixel state spaces such as Atari games. This would, in particular, imply adapting existing pseudo-counts techniques to the batch setting.

References

[1] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*, 2016.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[3] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[4] Erick Delage and Shie Mannor. Percentile optimization for markov decision processes with parameter uncertainty. *Operations research*, 2010.

[5] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1329–1338, 2016.

[6] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[7] Lior Fox, Leshem Choshen, and Yonatan Loewenstein. Dora the explorer: Directed outreaching reinforcement action-selection. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[8] Aude Genevay and Romain Laroche. Transfer learning for user adaptation in spoken dialogue systems. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2016.

[9] Ronald A Howard. Dynamic programming. *Management Science*, 1966.

[10] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*. Springer, 2012.

[11] Romain Laroche and Raphaël Féraud. Reinforcement learning algorithm selection. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[12] Romain Laroche and Paul Trichelair. Safe policy improvement with baseline bootstrapping. In *Proceedings of the 14th European Workshop on Reinforcement Learning*, 2018.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[14] Marek Petrik, Mohammad Ghavamzadeh, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*, 2016.

[15] Martin L Puterman and Shelby L Brumelle. On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 1979.

[16] Thiago D. Simao and Matthijs T. J. Spaan. Safe policy improvement with baseline bootstrapping in factored environments. In *Proceedings of the 33th AAAI Conference on Artificial Intelligence*, 2019.

[17] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[18] István Szita and András Lőrincz. The many faces of optimism: a unifying approach. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

[19] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[20] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

Batch Policy Learning under Constraints

Hoang M. Le
Department of Computing + Mathematical Sciences
California Institute of Technology
Pasadena, CA 91106
hmle@caltech.edu

Cameron Voloshin
Department of Computing + Mathematical Sciences
California Institute of Technology
Pasadena, CA 91106
clvoloshin@gmail.com

Yisong Yue
Department of Computing + Mathematical Sciences
California Institute of Technology
Pasadena, CA 91106
yyue@caltech.edu

Abstract

When learning policies for real-world domains, two important questions arise: (i) how to efficiently use pre-collected off-policy, non-optimal behavior data; and (ii) how to mediate among different competing objectives and constraints. We thus study the problem of batch policy learning under multiple constraints, and offer a systematic solution. We first propose a flexible meta-algorithm that admits any batch reinforcement learning and online learning procedure as subroutines. We then present a specific algorithmic instantiation and provide performance guarantees for the main objective and all constraints. To certify constraint satisfaction, we propose a new and simple method for off-policy policy evaluation (OPE) and derive PAC-style bounds. Our algorithm achieves strong empirical results in different domains, including in a challenging problem of simulated car driving subject to multiple constraints such as lane keeping and smooth driving. We also show experimentally that our OPE method outperforms other popular OPE techniques on a standalone basis, especially in a high-dimensional setting.

Keywords: Batch Reinforcement Learning
Constrained Policy Learning
Off-Policy Policy Evaluation
Off-Policy Learning
Approximate Dynamic Programming

1 Introduction

We study the problem of policy learning under multiple constraints. Contemporary approaches to learning sequential decision making policies have largely focused on optimizing some cost objective that is easily reducible to a scalar value function. However, in many real-world domains, choosing the right cost function to optimize is often not a straightforward task. Frequently, the agent designer faces multiple competing objectives. Indeed, many such real-world applications require the primary objective function be augmented with an appropriate set of constraints.

Recent policy learning research has largely focused on either online reinforcement learning (RL) with a focus on exploration, or imitation learning (IL) with a focus on learning from expert demonstrations. However, many real-world settings already contain large amounts of pre-collected data generated by existing policies (e.g., existing driving behavior, power grid control policies, etc.). We thus study the complementary question: *can we leverage this abundant source of (non-optimal) behavior data in order to learn sequential decision making policies with provable guarantees on constraint satisfaction*?

We present an algorithmic framework for learning sequential decision making policies from off-policy data. We employ multiple learning reductions to online and supervised learning, and present an analysis that relates performance in the reduced procedures to the overall performance with respect to both the primary objective and constraint satisfaction.

Constrained optimization is a well studied problem in supervised machine learning and optimization. In contrast to supervised learning for classification, batch policy learning for sequential decision making introduces multiple additional challenges. First, setting aside the constraints, batch policy learning itself presents a layer of difficulty, and the analysis is significantly more complicated. Second, verifying whether the constraints are satisfied is no longer as straightforward as passing the training data through the learned classifier. Certifying constraint satisfaction amounts to an off-policy policy evaluation problem, which by itself is a challenging problem. In this paper, we develop a systematic approach to address these challenges, provide a careful error analysis, and experimentally validate our algorithms.

2 Problem Formulation

Let $\mathrm{X} \subset \mathbb{R}^d$ be a bounded and closed d-dimensional state space. Let A be a finite action space. Let $c : \mathrm{X} \times \mathrm{A} \mapsto [0, \overline{C}]$ be the primary objective cost function that is bounded by $\overline{C}$. Let there be m constraint cost functions, $g_i : \mathrm{X} \times \mathrm{A} \mapsto [0, \overline{G}]$, each bounded by $\overline{G}$. To simplify the notation, we view the set of constraints as a vector function $g : \mathrm{X} \times \mathrm{A} \mapsto [0, \overline{G}]^m$ where $g(x, a)$ is the column vector of individual $g_i(x, a)$. Let $p(\cdot|x, a)$ denote the (unknown) transition/dynamics model. Let $\gamma \in (0, 1)$ denote the discount factor. Let χ be the initial states distribution.

In the discounted infinite horizon setting, an MDP is defined as $(\mathrm{X}, \mathrm{A}, c, g, p, \gamma, \chi)$. A policy $\pi \in \Pi$ maps states to actions, i.e., $\pi(x) \in \mathrm{A}$. The value function $C^\pi : \mathrm{X} \mapsto \mathbb{R}$ corresponding to the primary cost function c is defined in the usual way: $C^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(x_t, a_t) \mid x_0 = x\right]$. We similarly define the vector-value function for the constraint costs $G^\pi : \mathrm{X} \mapsto \mathbb{R}^m$ as $G^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t g(x_t, a_t) | x_0 = x\right]$. Let $C(\pi)$ and $G(\pi)$ as the expectation of $C^\pi(x)$ and $G^\pi(x)$, respectively.

In batch policy learning, we have a pre-collected dataset, $\mathrm{D} = \{(x_i, a_i, x'_i, c(x_i, a_i), g_{1:m}(x_i, a_i)\}_{i=1}^n$, generated from (a set of) historical behavioral policies denoted jointly by π_{D}. The goal of batch policy learning under constraints is to learn a policy $\pi \in \Pi$ from D that minimizes the primary objective cost while satisfying m different constraints:

$$\begin{aligned} \min_{\pi \in \Pi} \quad & C(\pi) \\ \text{s.t.} \quad & G(\pi) \leq \tau \end{aligned} \tag{OPT}$$

where $G(\cdot) = [g_1(\cdot), \ldots, g_m(\cdot)]^\top$ and $\tau \in \mathbb{R}^m$ is a vector of known constants. We assume that (OPT) is feasible. However, the dataset D might be generated from multiple policies that violate the constraints.

Example: Counterfactual & Safe Policy Learning. In conventional online RL, the agent needs to "re-learn" from scratch when the cost function is modified. Our framework enables counterfactual policy learning assuming the ability to compute the new cost objective from the same historical data. A simple example is *safe* policy learning. The goal here is to counterfactually avoid undesirable behaviors observed from historical data.

Example: Multi-objective Batch Learning. Traditional policy learning (RL or IL) presupposes that the agent optimizes a single cost function. In reality, we may want to satisfy multiple objectives that are not easily reducible to a scalar objective function. One example is learning fast driving policies under multiple behavioral constraints such as smooth driving and lane keeping consistency (see Section 5).

3 Proposed Approach

We first *convexify* the policy class Π by allowing stochastic combinations of policies, which effectively expands Π into its convex hull $\texttt{Conv}(\Pi)$. Formally, $\texttt{Conv}(\Pi)$ contains *randomized policies*.[1] Executing a mixed π consists of first sampling *one* policy π_t, and then executing π_t. It is easy to see that the augmented version of (OPT) over $\texttt{Conv}(\Pi)$ has a solution at least as good as the original (OPT).

[1] This places no restrictions on the individual policies. Individual policies can be arbitrarily non-convex. Convexifiying a policy class amounts to allowing ensembles of learned policies.

A Meta-Algorithm. The Lagrangian of (OPT) is $L(\pi, \lambda) = C(\pi) + \lambda^\top (G(\pi) - \tau)$ for $\lambda \in \mathbb{R}^m_+$. Clearly (OPT) is equivalent to the min-max problem: $\min_{\pi \in \Pi} \max_{\lambda \in \mathbb{R}^k_+} L(\pi, \lambda)$. We assume (OPT) is feasible. Policy class convexification ensure that strong duality holds, and (OPT) is also equivalent to: $\max_{\lambda \in \mathbb{R}^k_+} \min_{\pi \in \Pi} L(\pi, \lambda)$. From a game-thoeretic perspective, the problem becomes finding the equilibrium of a two-player game between the $\pi-$player and the $\lambda-$player [4]. In this repeated game, the $\pi-$player minimizes $L(\pi, \lambda)$ given the current λ, and the $\lambda-$player maximizes it given the current π.

Algorithm 1 Meta-algorithm

1: **for** each round t **do**
2: $\quad \pi_t \leftarrow$ `Best-response`(λ_t)
3: $\quad \widehat{\pi}_t \leftarrow \frac{1}{t}\sum_{t'=1}^t \pi_{t'}$, $\widehat{\lambda}_t \leftarrow \frac{1}{t}\sum_{t'=1}^t \lambda_{t'}$
4: $\quad \mathrm{L}_{\max} = \max_\lambda L(\widehat{\pi}_t, \lambda)$
5: $\quad \mathrm{L}_{\min} = L($`Best-response`$(\widehat{\lambda}_t), \widehat{\lambda}_t)$
6: $\quad$ **if** $\mathrm{L}_{\max} - \mathrm{L}_{\min} \le \omega$ **then**
7: $\quad\quad$ Return $\widehat{\pi}_t$
8: $\quad \lambda_{t+1} \leftarrow$ `Online-algorithm`$(\pi_1, \ldots, \pi_{t-1}, \pi_t)$

Algorithm 3 Fitted Q Evaluation `FQE`(π, c)

Input: Dataset $\mathrm{D} = \{x_i, a_i, x'_i, c_i\}_{i=1}^n \sim \pi_{\mathrm{D}}$
1: Initialize $Q_0 \in \mathrm{F}$ randomly
2: **for** $k = 1, 2, \ldots, K$ **do**
3: $\quad$ Compute target $y_i = c_i + \gamma Q_{k-1}(x'_i, \pi(x'_i)) \ \ \forall i$
4: $\quad$ Build training set $\widetilde{\mathrm{D}}_k = \{(x_i, a_i), y_i\}_{i=1}^n$
5: $\quad$ Solve a supervised learning problem:
$$Q_k = \arg\min_{f \in \mathrm{F}} \tfrac{1}{n}\textstyle\sum_{i=1}^n (f(x_i, a_i) - y_i)^2$$
Output: $\widehat{C}^\pi(x) = Q_K(x, \pi(x)) \quad \forall x$

Algorithm 2 Constrained Batch Policy Learning

Input: Dataset $\mathrm{D} = \{x_i, a_i, x'_i, c_i, g_i\}_{i=1}^n \sim \pi_{\mathrm{D}}$
1: Initialize $\lambda_1 = (\frac{B}{m+1}, \ldots, \frac{B}{m+1}) \in \mathbb{R}^{m+1}$
2: **for** each round t **do**
3: $\quad$ Learn $\pi_t \leftarrow$ `FQI`$(c + \lambda_t^\top g)$
4: $\quad$ Evaluate $\widehat{C}(\pi_t) \leftarrow$ `FQE`(π_t, c)
5: $\quad$ Evaluate $\widehat{G}(\pi_t) \leftarrow$ `FQE`(π_t, g)
6: $\quad \widehat{\pi}_t \leftarrow \frac{1}{t}\sum_{t'=1}^t \pi_{t'}$
7: $\quad \widehat{C}(\widehat{\pi}_t) \leftarrow \frac{1}{t}\sum_{t'=1}^t \widehat{C}(\pi_{t'})$, $\widehat{G}(\widehat{\pi}_t) \leftarrow \frac{1}{t}\sum_{t'=1}^t \widehat{G}(\pi_{t'})$
8: $\quad \widehat{\lambda}_t \leftarrow \frac{1}{t}\sum_{t'=1}^t \lambda_{t'}$
9: $\quad$ Learn $\widetilde{\pi} \leftarrow$ `FQI`$(c + \widehat{\lambda}_t^\top g)$
10: $\quad$ Evaluate $\widehat{C}(\widetilde{\pi}) \leftarrow$ `FQE`$(\widetilde{\pi}, c)$, $\widehat{G}(\widetilde{\pi}) \leftarrow$ `FQE`$(\widetilde{\pi}, g)$
11: $\quad \widehat{\mathrm{L}}_{\max} = \max_{\lambda, \|\lambda\|_1 = B} \left(\widehat{C}(\widehat{\pi}_t) + \lambda^\top \left[(\widehat{G}(\widehat{\pi}_t) - \tau)^\top, 0 \right]^\top \right)$
12: $\quad \widehat{\mathrm{L}}_{\min} = \widehat{C}(\widetilde{\pi}) + \widehat{\lambda}_t^\top \left[(\widehat{G}(\widetilde{\pi}) - \tau)^\top, 0 \right]^\top$
13: $\quad$ **if** $\widehat{\mathrm{L}}_{\max} - \widehat{\mathrm{L}}_{\min} \le \omega$ **then**
14: $\quad\quad$ Return $\widehat{\pi}_t$
15: $\quad$ Set $z_t = \left[(\widehat{G}(\pi_t) - \tau)^\top, 0 \right]^\top \in \mathbb{R}^{m+1}$
16: $\quad \lambda_{t+1}[i] = B \frac{\lambda_t[i] e^{-\eta z_t[i]}}{\sum_j \lambda_t[j] e^{-\eta z_t[j]}} \forall i$

We first present a meta-algorithm (Algorithm 1) that uses any no-regret online learning algorithm (for λ) and batch policy optimization (for π). At each iteration, given λ_t, the π-player runs `Best-response` to get the best response:

$$\texttt{Best-response}(\lambda_t) = \arg\min_{\pi \in \Pi} L(\pi, \lambda_t) = \arg\min_{\pi \in \Pi} C(\pi) + \lambda_t^\top (G(\pi) - \tau).$$

This is equivalent to a standard batch reinforcement learning problem where we learn a policy that is optimal with respect to $c + \lambda_t^\top g$. The corresponding mixed strategy is the uniform distribution over all previous π_t. In response to the $\pi-$player, the $\lambda-$player employs `Online-algorithm`, which can be *any* no-regret algorithm that satisfies: $\sum_t L(\pi_t, \lambda_t) \ge \max_\lambda \sum_t L(\pi_t, \lambda) - o(T)$. Finally, the algorithm terminates when the estimated primal-dual gap is below a threshold ω (Lines 7-8). Leaving aside (for now) issues of generalization, Algorithm 1 is guaranteed to converge assuming: (i) `Best-response` gives the best policy in the class, and (ii) $\mathrm{L}_{\max}$ and $\mathrm{L}_{\min}$ can be evaluated exactly.

Proposition 3.1. *Assuming (i) and (ii) above, Algo 1 is guaranteed to stop and the rate depends on the regret of* `Online-algorithm`*.*

3.1 Our Main Algorithm

We now provide a specific instantiation of Algorithm 1. Algorithm 2 is our main algorithm in this paper.

Policy Learning. We instantiate `Best-response` with Fitted Q Iteration (FQI), an off-policy learning approach [2].

Off-policy Policy Evaluation. A crucial difference between constrained policy learning and existing work on constrained supervised learning is the technical challenge of evaluating the objective and constraints. First, estimating $\widehat{L}(\pi, \lambda)$ (Lines 11-12) requires estimating $\widehat{C}(\pi)$ and $\widehat{G}(\pi)$. Second, any gradient-based approach to `Online-learning` requires passing in $\widehat{G}(\pi) - \tau$ as part of gradient estimate (line 15).

We propose a new and simple model-free technique using function approximation, called Fitted Q Evaluation (FQE). FQE is based on an iterative reductions scheme similar to FQI, but for the problem of off-policy evaluation. Algorithm 3 lays out the steps. The key difference with FQI is that the min operator is replaced by $Q_{k-1}(x'_i, \pi(x'_i))$ (Line 3 of Algorithm 3). Each x'_i comes from the original D. Since we know $\pi(x'_i)$, each $\widetilde{\mathrm{D}}_k$ is well-defined. Note that FQE can be plugged-in as a direct method if one wishes to augment the policy evaluation with a doubly-robust technique.

Online Learning Subroutine. Many online convex optimization approaches can be used for `Online-algorithm`. For our main Algorithm 2, we use Exponentiated Gradient (EG) [6], which has a regret bound of $O(\sqrt{\log(m)T})$ instead of $O(\sqrt{mT})$ as in OGD. radient-based algorithms generally require bounded λ. We thus force $\|\lambda\|_1 \le B$ using hyperparam-

eter B. Solving (OPT) exactly requires $B = \infty$. We will analyze Algorithm 2 with respect to finite B. We augment λ into a $(m+1)-$dimensional vector by appending $B - \|\lambda\|_1$, and augment the constraint cost vector g by appending 0.[2]

4 Theoretical Analysis

4.1 Convergence Guarantee

The convergence rate of Algorithm 2 depends on the radius B of the dual variables λ, the maximal constraint value $\overline{G}$, and the number of constraints m. In particular, we can show $O(\frac{B^2}{\omega^2})$ convergence for primal-dual gap ω.

Theorem 4.1 (Convergence of Algorithm 2)**.** *After T iterations, the empirical duality gap is bounded by*

$$\widehat{\mathrm{L}}_{\max} - \widehat{\mathrm{L}}_{\min} \leq 2\frac{B\log(m+1)}{\eta T} + 2\eta B\overline{G}^2$$

Thus, to achieve the primal-dual gap of ω, setting $\eta = \frac{\omega}{4\overline{G}^2 B}$ will ensure that Algo 2 converges after $\frac{16B^2\overline{G}^2\log(m+1)}{\omega^2}$ iterations.

4.2 Generalization Guarantee of FQE and FQI

We provide sample complexity analysis for FQE and FQI as *standalone* procedures for off-policy evaluation and off-policy learning. We use the notion of pseudo-dimension $\dim_{\mathrm{F}}$ as capacity measure of non-linear function class F [5]. Pseudo-dimension is finite for a large class of function approximators.

We use the notion of concentration coefficient, proposed by [8], to measure the degree of distribution shift. The following standard assumption from analysis of related ADP algorithms limits the severity of distribution shift over time:

Assumption 1 (Concentration coefficient of future state-action distribution)**.** *[9]*
Let P^π be the operator acting on $f : \mathrm{X}\times\mathrm{A} \mapsto \mathbb{R}$ s.t. $(P^\pi f)(x,a) = \int_{\mathrm{X}} f(x', \pi(x'))p(dx'|x,a)$. Given data generating distribution μ, initial state distribution χ, for $m \geq 0$ and an arbitrary sequence of stationary policies $\{\pi_m\}_{m\geq 1}$ define the concentration coeffient: $\beta_\mu(m) = \sup_{\pi_1,\ldots,\pi_m} \left\| \frac{d(\chi P^{\pi_1}P^{\pi_2}\ldots P^{\pi_m})}{d\mu} \right\|_\infty$. We assume $\beta_\mu = (1-\gamma)^2 \sum\limits_{m\geq 1} m\gamma^{m-1}\beta_\mu(m) < \infty$.

The performance of both FQE and FQI depend on how well the function class F approximates the Bellman operator. We measure the ability of function class F to approximate the Bellman evaluation operator via the worst-case Bellman error:

Definition 4.1 (inherent Bellman evaluation error)**.** Given a function class F and policy π, the *inherent Bellman evaluation error* of F is defined as $d_{\mathrm{F}}^\pi = \sup_{g\in\mathrm{F}} \inf_{f\in\mathrm{F}} \|f - \mathbb{T}^\pi g\|_\pi$ where $\|\cdot\|_\pi$ is the ℓ_2 norm weighted by distribution induced by π.

Theorem 4.2 (Generalization error of FQE)**.** *Under Assumption 1, for $\epsilon > 0$ & $\delta \in (0,1)$, after K iterations of Fitted Q Evaluation (Algorithm 3), for $n = O\big(\frac{\overline{C}^4}{\epsilon^2}(\log\frac{K}{\delta} + \dim_{\mathrm{F}}\log\frac{\overline{C}^2}{\epsilon^2} + \log\dim_{\mathrm{F}})\big)$, we have with probability $1-\delta$:*

$$\left|C(\pi) - \widehat{C}(\pi)\right| \leq \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\big(\sqrt{\beta_\mu}\,(2d_{\mathrm{F}}^\pi + \epsilon) + \frac{2\gamma^{K/2}\overline{C}}{(1-\gamma)^{1/2}}\big).$$

We can show an analogous generalization bound for FQI. While FQI has been widely used, to the best of our knowledge, a complete analysis of FQI for non-linear function approximation has not been previously reported.[3]

Definition 4.2 (inherent Bellman optimality error)**.** [9] Recall that the Bellman optimality operator is defined as $(\mathbb{T}Q)(x,a) = r(x,a) + \gamma\int_{\mathrm{X}} \min_{a'\in\mathrm{A}} Q(x',a')p(dx'|x,a)$. Given a function class F, the *inherent Bellman error* is defined as $d_{\mathrm{F}} = \sup_{g\in\mathrm{F}}\inf_{f\in\mathrm{F}}\|f - \mathbb{T}g\|_\mu$, where $\|\cdot\|_\mu$ is the ℓ_2 norm weighted by μ, the state-action distribution induced by π_{D}.

Theorem 4.3 (Generalization error of FQI)**.** *Under Assumption 1, for $\epsilon > 0$ & $\delta \in (0,1)$, after K iterations of Fitted Q Iteration, for $n = O\big(\frac{\overline{C}^4}{\epsilon^2}(\log\frac{K}{\delta} + \dim_{\mathrm{F}}\log\frac{\overline{C}^2}{\epsilon^2} + \log\dim_{\mathrm{F}})\big)$, we have with probability $1-\delta$:*

$$\left|C^* - C(\pi_K)\right| \leq \frac{2\gamma}{(1-\gamma)^3}\big(\sqrt{\beta_\mu}\,(2d_{\mathrm{F}} + \epsilon) + 2\gamma^{K/2}\overline{C}\big)$$

where π_K is the policy acting greedy with respect to the returned function Q_K.

4.3 End-to-End Generalization Guarantee

We are ultimately interested in the test-time performance and constraint satisfaction of the returned policy from Algorithm 2. We now connect the previous analyses from Theorems 4.1, 4.2 & 4.3 into an end-to-end error analysis.

Assumption 2. *Function classes* F *sufficiently rich so that $\forall f:\ \mathbb{T}f \in \mathrm{F}$ & $\mathbb{T}^\pi f \in \mathrm{F}$ for the policies π returned by Algorithm 2.*

Theorem 4.4 (Generalization guarantee of Algorithm 2)**.** *Let π^* be the optimal policy to* (OPT)*. Denote $\overline{V} = \overline{C} + B\overline{G}$. Let K be the number of iterations of FQE and FQI. Let $\widehat{\pi}$ be the policy returned by Algorithm 2, with termination threshold ω. For $\epsilon > 0$ & $\delta \in (0,1)$, when $n = O\big(\frac{\overline{V}^4}{\epsilon^2}(\log\frac{K(m+1)}{\delta} + \dim_{\mathrm{F}}\log\frac{\overline{V}^2}{\epsilon^2} + \log\dim_{\mathrm{F}})\big)$, we have with probability at least $1-\delta$:*

$$C(\widehat{\pi}) \leq C(\pi^*) + \omega + \frac{(4+B)\gamma}{(1-\gamma)^3}\big(\sqrt{\beta_\mu}\epsilon + 2\gamma^{K/2}\overline{V}\big),\ \text{and}\ G(\widehat{\pi}) \leq \tau + 2\frac{\overline{V}+\omega}{B} + \frac{\gamma^{1/2}}{(1-\gamma)^{3/2}}\big(\sqrt{\beta_\mu}\epsilon + \frac{2\gamma^{K/2}\overline{V}}{(1-\gamma)^{1/2}}\big).$$

[2] The $(m+1)^{th}$ coordinate of g is thus always satisfied. This augmentation is only necessary when executing EG.

[3] FQI for continuous action space from [1] is a variant of fitted policy iteration and not the version of FQI under consideration. The appendix of [7] contains a proof of FQI but for linear functions.

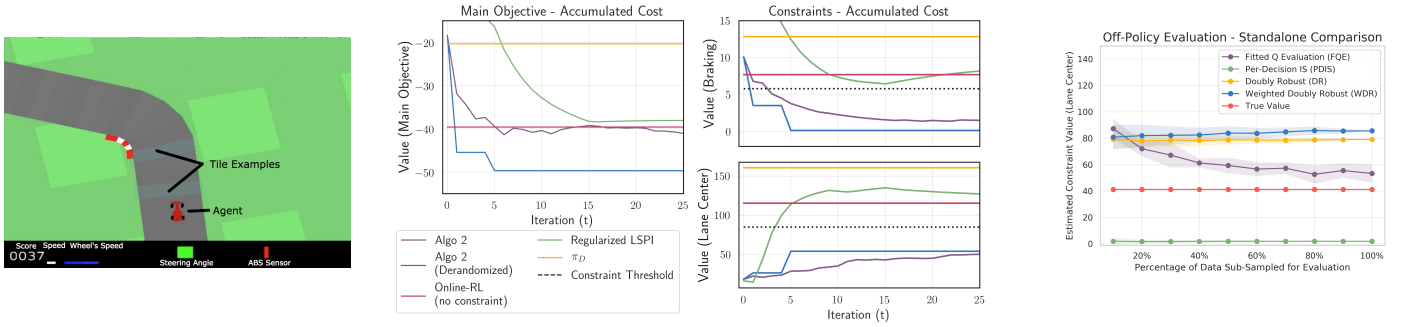


Figure 1: *CarRacing*. *(Left)* Screenshot of environment *(Middle)* (Lower is better) Comparing our algorithm, regularized LSPI, online RL w/o constraints, behavior policy π_D w.r.t. main cost objectives and two constraints. *(Right)* FQE vs. other OPE methods on a standalone basis.

5 Empirical Analysis - Car Racing under Smooth Driving and Lane Keeping Constraints

Environment & Data Collection. The car racing environment is a high-dimensional domain where the state is a $96 \times 96 \times 3$ tensor of raw pixels. The action space A takes 12 discretized values. The goal in this episodic task is to complete the track in minimum time. With the costs given by the environment, one can train online RL agent using DDQN. We collect ≈ 1500 trajectories from DDQN's randomization, resulting in data set D with ≈ 94000 transition tuples.

Fast Driving under Behavioral Constraints. We study the problem of minimizing cost while subject to two behavioral constraints: smooth driving and lane centering. This is a highly challenging setup since three objectives and constraints are in conflict with one another,e.g., fast driving causes the agent to cut corners and apply frequent brakes to make turns. We are not aware of previous work in policy learning with 2 or more constraints in high-dimensional settings.

Baseline and Procedure. As a naïve baseline, DDQN achieves low cost, but exhibits "non-smooth" driving behavior (see our supplementary videos). We set the threshold for each constraint to 75% of the DDQN benchmark. We also compare against regularized batch RL algorithms [3], specifically regularized LSPI. We instantiate our subroutines, FQE and FQI, with multi-layered CNNs. We augment LSPI's linear policy with non-linear features derived from a FQI model.

Results. The returned mixture policy from our algorithm achieves low main objective cost, comparable with online RL policy trained without regard to constraints. After several initial iterations violating the braking constraint, the returned policy - corresponding to the appropriate λ trade-off - satisties both constraints, while improving the main objective. The improvement over data gathering policy is significant for both constraints and main objective. While regularized LSPI obtains good performance for the main objective, it does not achieve acceptable constraint satisfaction. By default, regularized policy learning requires parameter tuning heuristics. In constrast, our approach is systematic, and is able to avoid the curse-of-dimensionality of brute-force search that comes with multiple constraints.

The off-policy evaluation by FQE is critical for success of our algorithm, and is ultimately responsible for certifying constraint satisfaction. While other OPE methods can also be used in place of FQE, we find that the estimates from popular methods are not sufficiently accurate in a high-dimensional setting. As a standalone comparison, we select an individual policy and compare FQE against PDIS, DR and WDR with respect to the constraint cost evaluation. To compare both accuracy and data-efficiency, for each domain we randomly sample different subsets of dataset D. Figure 1 (right) illustrates the difference in quality. In the high-dimensional car domain, FQE signficantly outperforms other methods.

References

[1] A. Antos, C. Szepesvári, and R. Munos. Fitted q-iteration in continuous action-space mdps. In *Advances in neural information processing systems*, pages 9–16, 2008.

[2] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr): 503–556, 2005.

[3] A. M. Farahmand, M. Ghavamzadeh, S. Mannor, and C. Szepesvári. Regularized policy iteration. In *Advances in Neural Information Processing Systems*, pages 441–448, 2009.

[4] Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

[5] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Springer, 2001.

[6] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[7] A. Lazaric and M. Restelli. Transfer from multiple mdps. In *Advances in Neural Information Processing Systems*, pages 1746–1754, 2011.

[8] R. Munos. Error bounds for approximate policy iteration. In *ICML*, volume 3, pages 560–567, 2003.

[9] R. Munos and C. Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857, 2008.

Model-based Knowledge Representations

Lucas Lehnert
Computer Science Department
Brown University
Providence, RI 02912, USA
lucas_lehnert@brown.edu

Michael L. Littman
Computer Science Department
Brown University
Providence, RI 02912, USA
michael_littman@brown.edu

Michael J. Frank
Department of Cognitive, Linguistic & Psychological Sciences
Carney Institute for Brain Science
Brown University
Providence, RI 02912, USA
michael_frank@brown.edu

Abstract

One question central to reinforcement learning is which representations – including aspects of the state space, transition function and reward function – can be generalized or re-used across different tasks. Humans are adept at such flexible transfer but existing reinforcement learning algorithms are much more limited. While transferring successor features between different tasks has been shown to improve learning speed, this representation is overly specific and hence needs to be re-learned when the optimal policy or transition function change.

This article presents Model Features: a latent representation that compresses the state space of a control problem by exploiting states that are equivalent in terms of both transition and reward functions. Because Model Features only extract these equivalences but are not tied to the transition and reward functions themselves, this latent state representation generalizes across tasks that change in both their transition and reward functions. Model Features link successor features to model reductions, facilitating the design of gradient-based optimization algorithms to approximate model reductions directly from transition data. Learning Model Features is akin to model-based reinforcement learning, because the learned representation supports predictions of future reward outcomes.

This article first summarizes theoretical results from our extended article. Then empirical simulation results are presented that suggest Model Features serve as a state representation that affords generalization across tasks with different transition and reward functions. Because Model Features construct a latent state representation that supports predictions of future reward outcomes, the presented results motivate further experiments to investigate if humans or animals learn such a representation, and whether neural systems involved in state representation reflect the equivalence abstraction.

Keywords: Model-based Reinforcement Learning, Knowledge Representations, Latent Structure Learning, Successor Representation, Human and Animal Learning

Acknowledgements

This work has been supported by funding from the MURI PERISCOPE project.

1 Introduction

Reinforcement learning (RL) [19] studies the problem of computing optimal decision strategies from one-step interactions sampled from an environment. Each interaction consists of the agent selecting an action to cause a change in the environment's state. For each state transition the agent receives a reward, a single scalar number. The goal is to compute a decision making strategy that maximizes rewards. One central question is how to generalize knowledge across different environments. This article presents Model Features, a feature representation that compresses the state space into a lower dimensional representation by clustering states that produce identical future reward outcomes. By only exploiting such state equivalences, Model Features generalize across environments that differ in reward and transition functions.

In RL a task is formalized as a Markov decision processes (MDP) [19] $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$. All possible states of a task are described by the set $\mathcal{S}$ and the agent can make a decision by selecting an action from a set of actions $\mathcal{A}$. Selecting an action changes the state probabilistically according to the transition function p. The reward function r specifies a reward for each state transition. The goal is to compute a decision strategy, called policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes rewards. One key property of model-based RL is the ability to predict a sequence of reward outcomes $(r_1, r_2, ...)$ given a state s and a sequence of actions $(a_1, a_2, ...)$. Model Features compress the state space by assigning two different states (approximately) the same n-dimensional feature vector if, given an arbitrary action sequence $(a_1, a_2, ...)$, both states generate the same reward sequence $(r_1, r_2, ...)$ with equal (or near-equal) probability. Such two states are also called bisimilar [10]. Knowing which reward sequence a certain action sequence can generate is sufficient for evaluating any arbitrary policy and identifying the optimal policy. Model Features encode model reductions [10] and provide a state representation specifically optimized to preserve all information relevant for predicting future reward outcomes.

This article first summarizes results from our extended article [12] and shows that Model Features, and by extension bisimulation relations, can be extracted by learning the successor features [2] of a single arbitrary policy. Model Features construct a low dimensional representation of the state space by utilizing which states are equivalent to the transition and reward function. We then show that Model Features learned for one MDP can be re-used on another MDP with different transition and reward functions, assuming that state equivalences are preserved. Such "deep transfer" across environments, even in the absence of prior experience with specific transition or reward functions, is predicted by behavioral and neural signatures of human structure learning [3, 1, 9] but not afforded by alternative algorithms that compress the transition function directly [18, 17]. As a lay example, an expert musician can immediately transfer a learned song from one key to another, or from a guitar to a piano, despite the very different transition functions [8].

2 Successor Features encode Model Features

A state representation ϕ maps each state s of an MDP to a real valued vector $\boldsymbol{\phi}$. Using ϕ a trajectory $(s_1, a_1, s_2, a_2, ...)$ can be mapped to a feature-trajectory $(\boldsymbol{\phi}_1, a_1, \boldsymbol{\phi}_2, a_2, ...)$. Because ϕ is a many-to-one function, different states may not be distinguishable in terms of their feature vectors. As a result, the empirical probability of transitioning between two different feature vectors need not correspond to the probability of the underlying state transition. Model Features are designed to distinguish between states such that the resulting empirical feature transition probabilities are equal to the underlying transition probabilities. A linear model can be build to predict feature-trajectories and reward sequences from these feature trajectories. If these predicted reward sequences equal in expectation the reward sequence $(r_1, r_2, ...)$ generated by the trajectory $(s_1, a_1, s_2, a_2, ...)$, then the state representation ϕ is said to support accurate predictions of future reward outcomes. Model Features are state representation that support predictions of future reward outcomes.

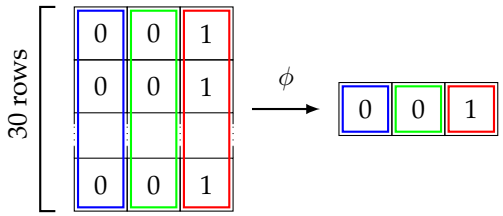
Suppose a state representation ϕ maps the state space to a finite set of feature vectors $\{\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_n\}$. By simulating different trajectories $(s_1, a_1, s_2, a_2, ...)$, the empirical probability $\Pr\{\boldsymbol{\phi}_i \xrightarrow{a} \boldsymbol{\phi}_j\}$ of transitioning from the feature vector $\boldsymbol{\phi}_i$ to $\boldsymbol{\phi}_j$ when selecting action a can be computed for all $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$. Figure 1(b) illustrates how a state representation partitions the state space by mapping different states to the same feature vector. Each state partition is a sub-set of the state space that corresponds to a particular feature vector $\boldsymbol{\phi}_i$. Suppose s is a state that is mapped to $\boldsymbol{\phi}_i$. If the probability $\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_j\}$ of transitioning from state s to the state partition corresponding to $\boldsymbol{\phi}_j$ is equal for all states that map to $\boldsymbol{\phi}_i$, then

$$\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_j\} = \Pr\{\boldsymbol{\phi}_i \xrightarrow{a} \boldsymbol{\phi}_j\} \text{ where } \phi(s) = \boldsymbol{\phi}_i. \tag{1}$$

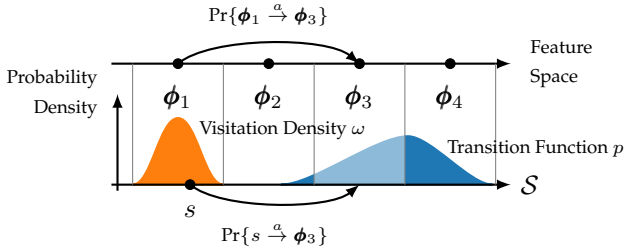
Line (1) holds because the empirical probability $\Pr\{\boldsymbol{\phi}_i \xrightarrow{a} \boldsymbol{\phi}_j\}$ is the marginal over all states mapping to $\boldsymbol{\phi}_i$:

$$\Pr\{\boldsymbol{\phi}_i \xrightarrow{a} \boldsymbol{\phi}_j\} = \int_{s_i : \phi(s) = \boldsymbol{\phi}_i} \omega(s_i) \Pr\{s_i \xrightarrow{a} \boldsymbol{\phi}_j\} ds = \left(\int_{s_i : \phi(s) = \boldsymbol{\phi}_i} \omega(s_i) ds \right) \Pr\{s \xrightarrow{a} \boldsymbol{\phi}_j\} = \Pr\{s \xrightarrow{a} \boldsymbol{\phi}_j\}. \tag{2}$$

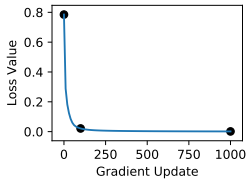
The empirical feature transition probability $\Pr\{\boldsymbol{\phi}_i \xrightarrow{a} \boldsymbol{\phi}_j\}$ is the marginal over all states s that map to $\boldsymbol{\phi}_i$ with respect to probability of visiting a state s. The density function ω models this visitation distribution over a state partition (Figure 1(b)). If $\Pr\{s_i \xrightarrow{a} \boldsymbol{\phi}_j\}$ is equal for all states s_i of the same partition, the probability of transitioning from state s to a state partition corresponding to $\boldsymbol{\phi}_j$ can be moved out of the integrand and the second identity in line (2) follows.



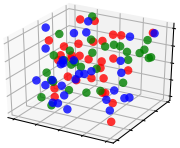
(a) Column World. To support predictions of future rewards, each column can be compressed to a state.



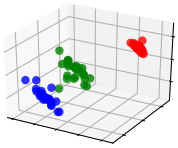
(b) Relationship between state transition probabilities and empirical feature transition probabilities.



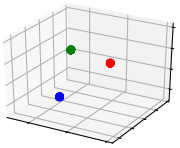
(c) Loss Objective [12]



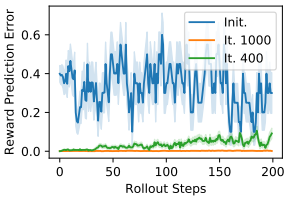
(d) Initialization



(e) Iteration 100



(f) Iteration 1000



(g) Reward Prediction Error

Figure 1: Learning Model Features. Figure 1(a) shows the Column World example where an agent can move up, down, left, or right (four actions) and is rewarded in the right column. Figure 1(b) shows the connection between empirical feature transition probabilities and state transition probabilities. Figures 1(c) to 1(f) plot how the loss objective evolves during optimization for the column world MDP. As the value of the loss objective decreases, the feature vectors for equivalent states merge into the same cluster and the different cluster centers move apart. At different training iterations, Figure 1(g) shows the average reward prediction errors along randomly selected 200-step action sequences and start states. Each curve plots the average over 100 action sequences and the shaded areas show standard errors.

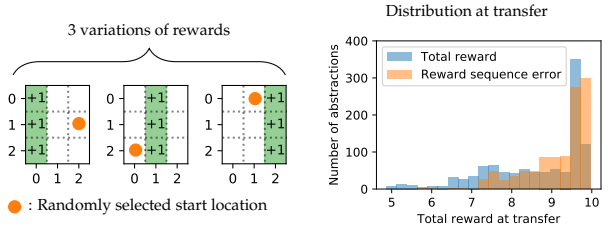
Clustering bisimilar states by directly comparing empirical feature transition probabilities to state transition probabilities is possible [10, 16], but such methods are restricted to constructing entire transition tables. By tying Model Features to a variation of Successor Features (SFs) [2, 5], the usual SF learning algorithms can be used to approximate Model Features directly from sampled transitions. For an arbitrary fixed policy π, SFs are defined as

$$\boldsymbol{\psi}^{\pi}_{s,a} = \mathbb{E}\left[\boldsymbol{\phi}_1 + \gamma\boldsymbol{\phi}_2 + \gamma^2\boldsymbol{\phi}_3 + \cdots \middle| s = s_1, a = a_1, \pi\right] = \mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\boldsymbol{\phi}_t \middle| s = s_1, a = a_1, \pi\right]. \tag{3}$$

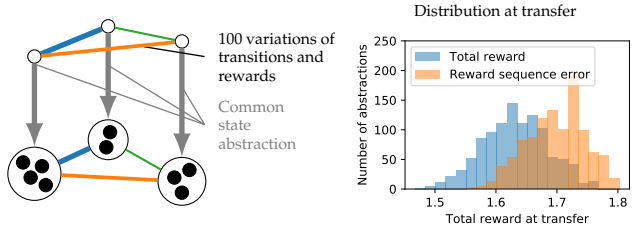
Because the expectation in Eq. (3) is computed over all possible infinite length trajectories starting at state s with action a, SFs implicitly encode the empirical feature transition probabilities for each action. Intuitively, SFs predict the frequencies or rescaled empirical probabilities with which feature vectors are encountered along a trajectory, because a frequently encountered feature vector will occur in the summation in Eq. (3) more often than rarely encountered feature vectors. If two states of the same state partition have the same SFs, then they also have the same state to state partition transition probabilities and Eq. (1) holds. Our extended article [12] formally proves that learning SFs is equivalent to finding approximations of Model Features. Figures 1(c) to 1(f) illustrate how learning SFs and minimizing the loss function plotted in Figure 1(c) results in increasingly accurate approximations of Model Features. Figure 1(g) displays the reward sequence prediction error at different steps of training. The curves show if reward prediction errors grow with the action sequence length. At initialization, reward prediction errors are low and as learning progresses, all reward prediction errors tend to zero. Hence, Model Features support predictions of future reward outcomes and model-based RL [12].

3 Model Features encode Task Knowledge

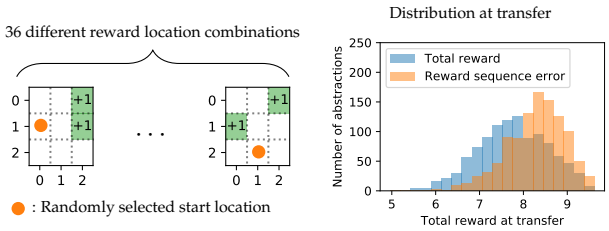
Model Features construct a low dimensional state representation by clustering states that are equivalent to the transition and reward function. On three transfer task sets, Figure 2 shows that a state representation's ability to predict future reward outcomes is indicative that this representation can be reused in a previously unseen MDP. On a previously unseen MDP such a representation performs better than other representations which only maximize total reward in the original MDP. If a state representation incorrectly clusters states, the resulting compressed MDP does not resemble the original MDP. A policy optimal in the compressed MDP is not necessarily optimal in the original MDP and cannot generate a high total reward. For each transfer experiment, all possible state representations were enumerated and scored on their reward sequence prediction error and the total reward generated. Total rewards were generated by a policy optimal in



(a) Transfer between different column worlds.



(b) Transfer between random MDPs compressible to 3 states.



(c) Transfer between grid worlds with different goal locations.

	p-value
Fig 2(a)	$3.29 \cdot 10^{-30}$
Fig 2(b)	$8.63 \cdot 10^{-176}$
Fig 2(c)	$9.60 \cdot 10^{-75}$

(d) One sided Welch's t-test.

Figure 2: Low reward sequence prediction errors identify state abstractions amenable for "deep transfer". For each experiment all possible state abstractions were enumerated using Algorithm U [11]. State abstractions were scored by compressing an MDP using the state abstraction of interest [14]. The total reward score was computed by solving for the optimal policy using value iteration [19, Chapter 4.4] and running the computed policy 20 times for 10 time steps in the MDP from a randomly selected start state. The reward sequence error was computed by selecting 20 random start states and then performing a random walk for 10 time steps. The histograms report averages over all repeats and transfer MDPs. Figure 2(d) lists the p-values of the difference in mean total reward being insignificant for each histogram.

the compressed version of one randomly selected MDP. The top 5% scoring state abstractions were then re-evaluated on the remaining transfer MDPs and the total rewards generated by these state representations are plotted as histograms in Figure 2. In all cases state representations with low reward sequence prediction errors generate a higher total reward at transfer than state representations that were selected based on their ability to construct a well performing policy on the original MDP. This result indicates that Model Features, which are designed to produce low reward sequence prediction errors, encode information about an MDP that generalizes across different MDPs.

Figure 2(a) presents three grid world MDPs where the agent can move up and down. The difference between the three MDPs is which column is rewarded. Each column can be compressed into a single state, similar to the column world example in Figure 1(a). For each experiment, total rewards and reward sequence prediction errors are computed by sampling a start location at random. The histogram in Figure 2(a) indicates that state representations with low reward sequence prediction errors outperform on average state representations that only maximize total reward in one of the tasks. The experiment in Figure 2(b) is similar to the previous experiment in that each of the 100 MDPs can be compressed with the same state representation, but otherwise the transition and reward functions are randomly generated. Besides a common "hidden" state representation, these 100 MDPs differ in both transition and reward functions. The histogram confirms the claim that low reward sequence prediction errors are indicative of a state representation's ability to generalize across different MDPs. State representations that result in high total reward in only one of the 100 MPDs generate on average less reward on any of the remaining MDPs. Figure 2(c) presents a transfer experiment where two reward locations are permuted in a grid world. In this experiment the MDPs cannot be compressed without incurring some loss, because the grid location is important for predicting where the goal locations are and what action is optimal at each location. However, the histogram in Figure 2(c) indicates that representations selected based on minimizing reward sequence prediction error criterion still perform better than selecting representations by their total reward. Because grid worlds have a specific topology of the state space, a state representation clustering only neighbouring states preserves approximately the grid location information and would be expected to perform relatively well across all MDPs.

4 Conclusion

Model Features approximate state abstractions that compress an MDP while preserving the ability to predict future reward outcomes using only the compressed model. Such a feature representation exploits which states are equivalent for both the reward and transition function and thus Model Features can be understood as a model-based knowledge repre-

sentation. Model-based basis functions have been studied previously [4, 6], but the presented connection to SFs allows us to design a gradient-based optimization algorithm that can construct bisimulation relations directly from transition data. Previous work on transfer with SFs has shown that re-using SFs on an MDP with a different reward function can provide faster learning [2, 15, 17]. However, SFs are fragile to changes in the optimal policy [13] and transition function, whereas latent state representations are more abstract and thus are not tied to particular transitions [3, 8]. In related work, Stachenfeld et al. [18] compress the SR of an MDP using PCA and demonstrate this representation's suitability for transfer and connections to place cells and grid cells in the hippocampus. However, this compressed SR constructs a representation of the transition function itself, and hence transfer is limited to environments that share the same transition function. In contrast to Stachenfeld et al., Model Features separate the transition dynamics (and the SR) from the compression on the state space itself, and thus generate a latent state representation of a task exploiting task equivalences. François-Lavet et al. [7] construct latent state representations as part of a model-free and model-based hybrid model. In contrast to their method, Model Features encode model reductions and are thus strict model-based representations. Collins and Frank [3] and Badre and Frank [1] demonstrate that reward prediction errors and prediction errors on the structure of the state space identify latent state representations that accelerate learning in humans when transferring knowledge across tasks. While this work considers contextual multi-armed bandits, Model Features may extend this work to sequential decision making, motivating further experiments to investigate if humans or animals learn such a feature representation.

References

[1] D. Badre and M. J. Frank. Mechanisms of hierarchical reinforcement learning in cortico–striatal circuits 2: Evidence from fmri. *Cerebral cortex*, 22(3):527–536, 2011.

[2] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[3] A. G. E. Collins and M. J. Frank. Neural signature of hierarchically structured expectations predicts clustering and transfer of rule sets in reinforcement learning. *Cognition*, 152:160–169, 2016.

[4] G. Comanici, D. Precup, and P. Panangaden. Basis refinement strategies for linear value function approximation in mdps. In *Advances in Neural Information Processing Systems*, pages 2899–2907, 2015.

[5] P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[6] N. Ferns and D. Precup. Bisimulation metrics are optimal value functions. In *UAI*, pages 210–219. Citeseer, 2014.

[7] V. François-Lavet, Y. Bengio, D. Precup, and J. Pineau. Combined reinforcement learning via abstract representations. *arXiv preprint arXiv:1809.04506*, 2018.

[8] N. T. Franklin and M. J. Frank. Compositional clustering in task structure learning. *PLoS computational biology*, 14 (4):e1006116, 2018.

[9] N. T. Franklin and M. J. Frank. Generalizing to generalize: when (and when not) to be compositional in task structure learning. *bioRxiv*, 2019. doi: 10.1101/547406.

[10] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1):163–223, 2003.

[11] D. E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions*. Addison-Wesley, 2005.

[12] L. Lehnert and M. L. Littman. Successor features support model-based and model-free reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2019.

[13] L. Lehnert, S. Tellex, and M. L. Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017.

[14] L. Li, T. J. Walsh, and M. L. Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

[15] I. Momennejad, E. M. Russek, J. H. Cheong, M. M. Botvinick, N. Daw, and S. J. Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680, 2017.

[16] S. S. Ruan, G. Comanici, P. Panangaden, and D. Precup. Representation discovery for mdps using bisimulation metrics. In *AAAI*, pages 3578–3584, 2015.

[17] E. M. Russek, I. Momennejad, M. M. Botvinick, S. J. Gershman, and N. D. Daw. Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS computational biology*, 13(9):e1005768, 2017.

[18] K. L. Stachenfeld, M. M. Botvinick, and S. J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20:1643 EP –, 10 2017. URL `https://doi.org/10.1038/nn.4650`.

[19] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Rethinking Expected Cumulative Reward Formalism of Reinforcement Learning: A Micro-Objective Perspective

Changjian Li
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON N2L3G1
changjian.li@uwaterloo.ca

Krzysztof Czarnecki
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON N2L3G1
k2czarne@uwaterloo.ca

Abstract

The standard *reinforcement learning* (RL) formulation considers the expectation of the (discounted) cumulative reward. This is limiting in applications where we are concerned with not only the *expected* performance, but also the distribution of the performance. In this paper, we introduce *micro-objective reinforcement learning* — an alternative RL formalism that overcomes this issue. In this new formulation, a RL task is specified by a set of *micro-objectives*, which are constructs that specify the desirability or undesirability of events. In addition, micro-objectives allow prior knowledge in the form of temporal abstraction to be incorporated into the global RL objective. The generality of this formalism, and its relations to single/multi-objective RL, and hierarchical RL are discussed.

Keywords: reinforcement learning; Markov decision process

Acknowledgements

The authors would like to thank Sean Sedwards, Jaeyoung Lee and other members of Waterloo Intelligent Systems Engineering Lab (WISELab) for discussions.

1 Introduction and Related Works

The RL formulation commonly adopted in literature aims to maximize the expected *return* (discounted cumulative reward), which is desirable if all we are concerned with is the expectation. However, in many practical problems, especially in risk-sensitive applications, we not only care about the expectation, but also the distribution of the return. For example, in autonomous driving, being able to drive well in expectation is not enough, we need to guarantee that the risk of collision is below a certain acceptable level. As another example, there might be two investment plans with the same expected return, but different variance. Depending on investor type, one investment plan might be more attractive than the other. As a simplified abstraction, we consider the following *Markov Decision Process* (MDP) with only one non-absorbing state s_0, which is the state where the investment decision is to be made. There are two actions, a_1 and a_2, corresponding to the two investment plans. From s_0, if a_1 is taken, there is 0.9 chance of getting a profit of 10 (entering absorbing state s_1), and 0.1 chance of getting a loss of -10 (entering absorbing state s_2). If a_2 is taken, there is 0.7 probability of earning a profit of 20 (entering absorbing state s_3), and 0.3 probability of receiving a loss of -20 (entering absorbing state s_4). The reward function is therefore as follows:

$$r(s_0, a_1, s_1) = 10, \quad r(s_0, a_1, s_2) = -10, \quad r(s_0, a_2, s_3) = 20, \quad r(s_0, a_2, s_4) = -20 \tag{1}$$

The reward function is zero onwards once an absorbing state is reached. Both a_1 and a_2 will result in an expected return of 8. However, the investor might not be able to afford a loss of more than, say, 15, in which case a_1 is preferable to a_2. Unfortunately, the expected return formulation provides no mechanism to differentiate these two actions. Furthermore, any mixture policy of a_1 and a_2 (mixing a_1 and a_2 with some probability) also has the same expected return.

Two approaches have been discussed in literature to tackle this issue. One is to shape the reward so that the expected return of the policies are no longer the same [Koenig and Simmons, 1994]. E.g., we can give more negative reward when the loss is higher than 15, such as the following:

$$r'(s, a, s') = \begin{cases} r(s, a, s'), & \text{if } r(s, a, s') \geq -15 \\ r(s, a, s') - (r(s, a, s') + 15)^2, & \text{otherwise} \end{cases}$$

Although in this simple case, the reward shaping is fairly straight-forward, in more complex tasks such as autonomous driving where there are many conflicting aspects, it is a challenge to choose a reward that properly balances the expected overall performance and the risk of each aspect.

The second approach is to use an alternative formulation that considers more than just the expected return. Several methods [Sato et al., 2001, Sherstan et al., 2018, Tamar et al., 2013] have been proposed to estimate the variance of return in addition to the expectation. While this alleviates the issue by taking variance into account, distributions can differ even if both the expectation and the variance are the same. Yu et al. [Yu et al., 1998] considered the problem of maximizing the probability of receiving a return that is greater than a certain threshold. Geibel and Wysotzki [Geibel and Wysotzki, 2011] considered constrained MDPs [Altman, 1999] where the discounted probabilities of error states (unacceptable states) are constrained. However, both of these formulations are designed only for a specific type of application, and do not have the generality required as an alternative RL formalism.

In this paper, we propose to solve this issue by restricting the return to a distribution that is entirely decided by its mean, namely the Bernoulli distribution. To motivate this idea, observe that in any RL task, we are essentially concerned with a set of events, some desirable, some undesirable, to different extents. For example, in the task of autonomous driving, collision is an undesirable event; running a red light is another event, still undesirable but not as much; driving within the speed limit is a desirable event, etc. Instead of associating each event with a reward, and evaluating a policy by the total reward it accumulates (as in conventional RL), we can think of all the events as a whole, and evaluate a policy based on the combination of events it would lead to. The return is now only an indicator of an event, and is restricted to binary values: 1 if the event happens, and 0 if it does not. Given a policy, the return of each micro-objective is thus a Bernoulli random variable, whose mean is both the *value function*, and the probability of event occurrence under the policy. Following this view, a task can be specified by a predefined set of events, and a partial order that allows comparison between different combinations of event probabilities. The goal is to find the policies that result in the most desirable combinations of probabilities. [1].

This can be illustrated with the investment example. Instead of defining a reward function as in Eq. 1, we define entering s_1, s_2, s_3 and s_4 as four events. If action a_1 is taken, the chances of the four events occurring are 0.9, 0.1, 0 and 0. If action a_2 is taken, the chances are $[0, 0, 0.7, 0.3]$. Apart from the events, a partial order on the probability vectors is also defined to specify which probability vector is more desirable. Let $\mathbf{v}^\pi = [v_1^\pi, v_2^\pi, v_3^\pi, v_4^\pi]$ denote the expected returns of the four events. If we set the partial order to $\mathbf{v}^\pi \preceq \mathbf{v}^{\pi'} \iff 20(v_3^\pi - v_4^\pi) + 10(v_1^\pi - v_2^\pi) \leq 20(v_3^{\pi'} - v_4^{\pi'}) + 10(v_1^{\pi'} - v_2^{\pi'})$, we arrive at an equivalent formulation to the standard RL formulation with reward specified by Eq. 1. If, however, we want the probability of getting a loss of more than 15 to be less than a certain threshold ϵ, we can simply redefine the partial order so that $\mathbf{v}^\pi$ is smaller whenever $v_4^\pi \geq \epsilon$.

[1] To be exact, the combinations (of probabilities) that are not less desirable than any other combinations.

2 Background

2.1 MDP and Reinforcement Learning

The standard RL problem is often formulated in terms of a (single-objective) *Markov Decision Process* (MDP), which can be represented by a six-tuple $(S, A, P, \mu, \gamma, r)$, where S is a finite set of states; A is a finite set of actions; $P(s'|s,a)$ is the transition probability from state s to state s' taking action a; μ is the initial state distribution; $\gamma \in [0,1]_R$ is the discount factor; and $r(s,a,s') \in R$ is the reward for taking action a in state s and arriving state s'. The goal is to find the maximal expected discounted cumulative reward: $\max_{\pi\in\Pi} E[\sum_{t=0}^{\infty} \gamma^{(t)} r(s^t, a^t, s^{t+1})|\pi, \mu]$, where $\gamma^{(t)}$ denotes γ to the power of t, and Π denotes the set of policies we would like to consider. In the most general case, a policy can be history-dependent and random, in the form of $\pi = (\pi^0, \pi^1, ..., \pi^t, ...)$, where a *decision rule* $\pi^t(h^t, s, a) \in [0,1]_R$ is the probability of taking action a in state s with history h^t. A history is a sequence of past states, actions and decision rules $h^t = (s^0, a^0, \pi^0, s^1, a^1, \pi^1, ..., s^{t-1}, a^{t-1}, \pi^{t-1})$. A policy is said be *deterministic* if $\pi^t(h^t, s, a) = 1$ for only one action, in which case we can use a simplified notation $\pi = (d^0, d^1, ..., d^t, ...)$, where $d^t(h^t, s) \in A$. Correspondingly, if the policy is deterministic, a history can be represented with $h^t = (s^0, d^0, s^1, d^1, ..., s^{t-1}, d^{t-1})$. A policy is said to be *stationary* if the decision rule only depends on the current state s, and does not change with time, i.e., $\pi^t(h^t, s, a) = \pi(s, a)$. The set of all history-dependent random policies, history-dependent deterministic policies, stationary random policies, and stationary deterministic policies are denoted by Π_{HR}, Π_{HD}, Π_{SR} and Π_{SD}, respectively. We call a task an *episodic* task with *horizon* T if the state space is augmented with time; $\gamma = 1$; and $r(s,a,s') = 0, \forall t \geq T$. The expected return following a policy starting from the initial state distribution is called the value function, which is denoted as v^π. For a single-objective MDP, there exists stationary deterministic optimal policy, that is, $\exists \pi \in \Pi_{\text{SD}}, v^\pi = \arg\max_{\pi' \in \Pi_{\text{HR}}} v^{\pi'}$

2.2 Multi-objective Reinforcement Learning

In some cases [Roijers et al., 2014], it is preferable to consider different aspects of a task as separate objectives. *Multi-objective reinforcement learning* is concerned with *multi-objective Markov decision processes* (MOMDPs) $(S, A, P, \mu, [(\gamma_1, r_1), ..., (\gamma_i, r_i), ..., (\gamma_k, r_k)], \preceq)$, where S, A and $P(s'|s,a)$, μ are the state space, action space, transition probability and initial state distribution as in single-objective MDPs; Now there are k pairs of discount factors γ_i and rewards $r_i(s,a,s')$, one for each objective. The value function for the i^{th} objective is defined as $v_i^\pi = E[\sum_{t=0}^{\infty} \gamma_i^{(t)} r_i(s^t, a^t, s^{t+1})|\pi, \mu]$. Let $\mathbf{v}^\pi = [v_1^\pi, v_2^\pi, ..., v_k^\pi]$ be the value functions for all objectives, and $V^\Pi = \{\mathbf{v}^\pi | \pi \in \Pi\}$ be the set of all realizable value functions by policies in Π, $\preceq$ is a partial order defined on $V^{\Pi_{\text{HR}}}$. Multi-objective RL aims to find the policies $\pi \in \Pi$ such that $\mathbf{v}^\pi$ is a *maximal element* [2] of V^Π. Episodic tasks have not been widely discussed in the context of multi-objective RL, and most literature assumes $\gamma_1 = \gamma_2 = ... = \gamma_k$. Although for a single-objective MDP, the optimal value can be attained by a deterministic stationary policy, this is in general not true for multi-objective MDPs. White [White, 1982] showed that history-dependent deterministic policies can dominate stationary deterministic policies; Chatterjee et al. [Chatterjee et al., 2006] proved for the case $\gamma_1 = \gamma_2 = ... = \gamma_k$ that there exists optimal stationary random policy.

3 Micro-Objective Reinforcement Learning

In standard multi-objective RL, there is no restriction on the reward function of each objective. Each objective can itself be a 'macro' objective that involves multiple aspects. This makes multi-objective RL subject to the same issue single-objective RL has: only the *expectation* of return is considered for each objective. Conceptually, micro-objective RL is multi-objective RL at its extreme: each *micro-objective* is concerned with one and only one aspect — the occurrence of an *event*. An event can be 'entering a set of goal/error states', 'taking certain actions in certain states', or 'entering a set of states at certain time steps', etc., but ultimately can be represented by a set of histories. If the history up to the current time step (h^t, s^t) is in the set, we say that the event happens.

3.1 Micro-objectives

At the core of the micro-objective formulation is a new form of value function $v^\pi_{\psi_i, T_i}(\phi_i|\mu)$. Denoting H as the set of all possible histories, $\psi_i \subset H$ is the set of histories that corresponds to the occurrence of the event, which we call the *termination set* of a micro-objective. $\phi_i \subset H$ is also a set of histories, which we call the *initiation set*. The terminologies are deliberately chosen to resemble those of *options* [Sutton et al., 1999], and as we will see, this form of value function is indeed connected to options. Independent from the task, a micro-objective has its own initiation and termination. A micro-objective initiates if it is not currently active and $(h^t, s^t) \in \phi_i$, when an associated timer t_i is also initiated. A micro-objective terminates if it is currently active and $(h^t, s^t) \in \psi_i$, upon which a return of 1 is received. It also terminates if $t_i \geq T_i$ or the task terminates, upon which a return of 0 is received, and t_i is reset. Note that t and T are the time step and time horizon for the task, whereas t_i and T_i are the time step and time horizon for the micro-objective. $v^\pi_{\psi_i, T_i}(\phi_i|\mu)$ is defined as the expected return the i^{th} micro-objective receives starting from initial state distribution μ following policy π. For example, suppose that the task always starts from s_0 (i.e., $\mu(s_0) = 1$), and the micro-objective is active

[2] A maximal element of a subset X of some partially ordered set is an element of X that is not smaller than any other element in X.

three times (in sequence) before task termination if policy π is followed, with the return of 0, 0 and 1, respectively, then $v^{\pi}_{\psi_i,T_i}(\phi_i|\mu)$ is $\frac{1}{3}(0+0+1)=\frac{1}{3}$. Although this particular form of value function is similar to the generalized value function (GVF) proposed by Sutton et al. [Sutton et al., 2011] in the sense that both have their own initiation, termination and return, it is a rather different concept. Unlike a GVF which is associated with a *target policy*, and can be interpreted as the answer to a question regarding the target policy; the value function of a micro-objective is parameterized by the global control policy, and is an evaluation of the global policy with respect to one aspect of the task. This becomes clearer when we consider the fact that the value function for a micro-objective is conditioned on the initial state distribution of the task. As a result, the value functions of the micro-objectives appear in the global RL objective, while it is not obvious how GVFs can be used in the RL task specification.

Formally, a micro-objective RL problem is *an episodic task* represented by the 8-tuple $(S, A, P, \mu, \psi, T, [(\phi_1, \psi_1, T_1), ..., (\phi_i, \psi_i, T_i), ..., (\phi_k, \psi_k, T_k)], \preceq)$, where S, A, P, μ are the state space, action space, transition probabilities, and initial state distribution as usual. $\psi \subset S$ is the terminal states of the task, and T is the time horizon. The task terminates whenever $t \geq T$, or a state in ψ is reached. (ϕ_1, ψ_1, T_1) to (ϕ_k, ψ_k, T_k) are the k micro-objectives as described above. Let $\mathbf{v}^{\pi} = [v^{\pi}_{\psi_1,T_1}(\phi_1|\mu), v^{\pi}_{\psi_2,T_2}(\phi_2|\mu), ..., v^{\pi}_{\psi_k,T_k}(\phi_k|\mu)]$ be the value functions for all objectives, and $V^{\Pi} = \{\mathbf{v}^{\pi}|\pi \in \Pi\}$ be the set of all realizable value functions by policies in Π, $\preceq$ is a partial order defined on $V^{\Pi_{\text{HR}}}$. Similar to multi-objective RL, the goal is to find the policies $\pi \in \Pi$ such that $\mathbf{v}^{\pi}$ is a maximal element of V^{Π}. However, the multi-objective RL formulation introduced in Section 2.2 does not subsume micro-objective RL. For one thing, there is no notion of objective termination in multi-objective RL. It can be shown that the optimal policy for micro-objective RL is in general history-dependent and random, which we omit due to the space limit.

3.2 Relation with Hierarchical RL

Hierarchical RL [Barto and Mahadevan, 2003] refers to RL paradigms that exploits *temporal abstraction* to facilitate learning, where the higher level policy selects 'macro' actions that in turn 'call' the lower level actions. Notable Hierarchical RL approaches include the *options* formalism [Sutton et al., 1999], *hierachical abstract machines* (HAM) [Parr and Russell, 1997], the MAXQ framework [Dietterich, 2000], and *feudal RL* [Dayan and Hinton, 1992]. However, in none of these frameworks does the specification for temporal abstraction appear in the global RL objective. As a result, there is no clear measure on how well each temporally abstracted action should be learned, and how important they are compared to the goal of the high-level task. The micro-objective formulation is designed to allow temporal abstraction to be expressed as micro-objectives, therefore building the bridge between 'objectives' and 'options'. To see this, recall that a micro-objective has an initiation set ϕ_i and a termination set ψ_i, corresponding to the initiation set $I \subset S$ and termination condition $\beta : S \to [0,1]_R$ of an option. If the initiation set I and the goal states [3] of an option coincide with the initiation set ϕ_i and the termination set ψ_i of a micro-objective, then the micro-objective can be thought of as a measure of how important this option is. To be more concrete, consider the following task where the initial state is s_0, the goal state is s_1, and ψ is a set of intermediate states. o_0 is a hypothetical (might not exist, and is to be learned if needed) option that takes the agent from s_0 to s_1 without passing through ψ. Similarly, o_1 is a hypothetical option from s_0 to ψ, and o_2 is a hypothetical option from ψ to s_1. Such a task can be a taxi agent at location s_0 driving a passenger from a pick-up location ψ to a destination s_1, in which case we can define two micro-objectives $(\{s_0\}, \psi, T_1)$ and $(\psi, \{s_1\}, T_2)$, corresponding to o_1 and o_2 respectively. In this example, a micro-objective for o_0 is not needed, because passing though ψ is required. The value function is thus $\mathbf{v}^{\pi} = [v^{\pi}_{\psi,T_1}(\{s_0\}|\mu(s_0)=1), v^{\pi}_{\{s_1\},T_2}(\psi|\mu(s_0)=1)]$, and the partial order can be defined as $\mathbf{v}^{\pi} \preceq \mathbf{v}^{\pi'} \iff v^{\pi}_{\psi,T_1}(\{s_0\}|\mu(s_0)=1) \leq v^{\pi'}_{\psi,T_1}(\{s_0\}|\mu(s_0)=1) \wedge v^{\pi}_{\{s_1\},T_2}(\psi|\mu(s_0)=1) \leq v^{\pi'}_{\{s_1\},T_2}(\psi|\mu(s_0)=1)$. Another possible scenario for such a task would be an agent navigating through a maze, and ψ is only some heuristics. In this case, the only important micro-objective is $(\{s_0\}, \{s_1\}, T_3)$, which corresponds to option o_3. o_1 and o_2 are only there to help exploration. Let $\mathbf{v}^{\pi} = [v^{\pi}_{\psi,T_1}(\{s_0\}|\mu(s_0)=1), v^{\pi}_{\{s_1\},T_2}(\psi|\mu(s_0)=1), v^{\pi}_{\{s_1\},T_3}(\{s_0\}|\mu(s_0)=1)]$, the partial order can be defined as

$$\mathbf{v}^{\pi} \preceq \mathbf{v}^{\pi'} \iff v^{\pi}_{\{s_1\},T_3}(\{s_0\}|\mu(s_0)=1) < v^{\pi'}_{\{s_1\},T_3}(\{s_0\}|\mu(s_0)=1) \vee \Big(v^{\pi}_{\{s_1\},T_3}(\{s_0\}|\mu(s_0)=1) = v^{\pi'}_{\{s_1\},T_3}(\{s_0\}|\mu(s_0)=1) \wedge$$
$$v^{\pi}_{\psi,T_1}(\{s_0\}|\mu(s_0)=1) \leq v^{\pi'}_{\psi,T_1}(\{s_0\}|\mu(s_0)=1) \wedge v^{\pi}_{\{s_1\},T_2}(\psi|\mu(s_0)=1) \leq v^{\pi'}_{\{s_1\},T_2}(\psi|\mu(s_0)=1) \Big)$$

3.3 Generality

We briefly discuss the generality of the micro-objective RL formulation. Since the state space, action space and transition probabilities are the same as in a MDP, our only concern is whether micro-objectives, together with the partial order can imply an arbitrary optimal policy. If for any stationary deterministic policy π^* and any Markov dynamics (S, A, P), there exists a partial order and a set of micro-objectives such that π^* is optimal, then we can cast any single-objective RL problem into an equivalent micro-objective problem. Now we show that micro-objective RL can indeed imply an arbitrary stationary deterministic policy $\pi^* \in \Pi_{\text{SD}}$. Let $s_1, s_2, ...s_{|S|}$ be an enumeration of the finite state space S, we define $|S|$ micro-objectives $v^{\pi}_{\{(h,s_j,\pi^*(s_j))|h\in H\},1}(\{(h,s_j)|h \in H\}|\mu), j = 1, 2, ..., |S|$, which, with abuse of notation, we write as v^{π}_j. In other words, for each state $s \in S$ we define an event: taking $\pi^*(s)$ in s. If we

[3] Options do not need to have goal states. This is an intentional oversimplification to illustrate the idea.

further define the partial order as $\mathbf{v}^{\pi} \preceq \mathbf{v}^{\pi'} \iff v_1^{\pi} \leq v_1^{\pi'} \wedge ... \wedge v_{|S|}^{\pi} \leq v_{|S|}^{\pi'}$, then π^* is optimal for this micro-objective RL task. Therefore, given enough micro-objectives, the micro-objective formulation is at least as general as the standard RL formulation.

4 Discussions

We introduced micro-objective RL, a general RL formalism that not only solves the problem of standard RL formulation that only the expectation is considered, but also allows temporal abstraction to be incorporated into the global RL objective. Intuitively, this micro-objective paradigm bears more resemblance to how humans perceive a task — it is hard for a human to tell what reward they receive at a certain time, but it is relatively easy to tell how good a particular combination of events is. Ongoing research topics include effective algorithms for micro-objective RL, and compact representation of similar micro-objectives.

References

Eitan Altman. *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999. ISBN 9780849303821.

Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2):41–77, 2003. doi: 10.1023/A:1022140919877. URL https://doi.org/10.1023/A:1022140919877.

Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov decision processes with multiple objectives. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 325–336, 2006. doi: 10.1007/11672142_26. URL https://doi.org/10.1007/11672142_26.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pages 271–278, 1992. URL http://papers.nips.cc/paper/714-feudal-reinforcement-learning.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Intell. Res.*, 13: 227–303, 2000. doi: 10.1613/jair.639. URL https://doi.org/10.1613/jair.639.

Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *CoRR*, abs/1109.2147, 2011. URL http://arxiv.org/abs/1109.2147.

Sven Koenig and Reid G. Simmons. Risk-sensitive planning with probabilistic decision graphs. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994.*, pages 363–373, 1994.

Ronald Parr and Stuart J. Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*, pages 1043–1049, 1997. URL http://papers.nips.cc/paper/1384-reinforcement-learning-with-hierarchies-of-machines.

Diederik Marijn Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *CoRR*, abs/1402.0590, 2014. URL http://arxiv.org/abs/1402.0590.

Makoto Sato, Hajime Kimura, and Shibenobu Kobayashi. Td algorithm for the variance of return and mean-variance reinforcement learning. *Transactions of the Japanese Society for Artificial Intelligence*, 16(3):353–362, 2001. doi: 10.1527/tjsai.16.353.

Craig Sherstan, Dylan R. Ashley, Brendan Bennett, Kenny Young, Adam White, Martha White, and Richard S. Sutton. Comparing direct and indirect temporal-difference methods for estimating the variance of the return. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 63–72, 2018. URL http://auai.org/uai2018/proceedings/papers/35.pdf.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, May 2-6, 2011, Volume 1-3*, pages 761–768, 2011. URL http://portal.acm.org/citation.cfm?id=2031726&CFID=54178199&CFTOKEN=61392764.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. Temporal difference methods for the variance of the reward to go. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 495–503, 2013. URL http://jmlr.org/proceedings/papers/v28/tamar13.html.

D.J White. Multi-objective infinite-horizon discounted markov decision processes. *Journal of Mathematical Analysis and Applications*, 89(2):639 – 647, 1982. ISSN 0022-247X. doi: https://doi.org/10.1016/0022-247X(82)90122-6. URL http://www.sciencedirect.com/science/article/pii/0022247X82901226.

Stella X Yu, Yuanlie Lin, and Pingfan Yan. Optimization models for the first arrival target distribution function in discrete time. *Journal of Mathematical Analysis and Applications*, 225(1):193 – 223, 1998. ISSN 0022-247X. doi: https://doi.org/10.1006/jmaa.1998.6015. URL http://www.sciencedirect.com/science/article/pii/S0022247X98960152.

Learning Treatment Policies for Mobile Health Using Randomized Least-Squares Value Iteration

Celine Liang
Harvard University
Cambridge, MA 02138
cliang@college.harvard.edu

Serena Yeung
Harvard University
Cambridge, MA 02138
serenayeung@g.harvard.edu

Susan Murphy
Harvard University
Cambridge, MA 02138
samurphy@fas.harvard.edu

Abstract

In this work, we investigate the use of Randomized Least-Squares Value Iteration (RLSVI), a recently proposed reinforcement learning algorithm, for learning treatment policies in mobile health. RLSVI uses a Bayesian approach to learn action-state value functions and then selects subsequent actions using the posterior distribution of this learned function. An important challenge in mobile health is to learn an optimal policy, that is, which treatment (usually in the form of a mobile notification) to provide a user in a given state. Providing too few notifications defeats the purpose of the intervention, while bombarding with too many notifications increases the burden of the user and reduces the effectiveness of the treatment. Learning a policy for a user is not easy and must be done efficiently to avoid user disengagement. To do this requires a delicate balance of exploration and exploitation. The goal of this research is to develop an online algorithm for mobile health that can learn and update the treatment policy efficiently and continuously. We consider policies for a binary action: sending a notification to the user (pinging) or refraining from any action (waiting). We first test the original, finite-horizon RLSVI algorithm on a testbed that reflects a simplified mobile health setting, for a two-dimensional state space over a finite time horizon. Notifying the user accumulates quantities of short-term reward at the expense of decreased potential total reward, while waiting accumulates no reward but increases the potential total reward in the future. We then develop and test a continuing task extension of RLSVI that is more relevant to mobile health problems in the real world and show that for both episodic and continuing tasks, RLSVI performs more robustly than an algorithm employing least-squares value iteration (LSVI) to learn the action-state value function while selecting actions in an epsilon-greedy manner.

Keywords: mobile health, user notification, treatment policies, randomized value iteration, continuing task

1 Introduction

With the rise in popularity of mobile devices and wearable technology, it is now possible to deliver behavioral treatments in situ (usually in the form of a mobile notification). In mobile health, one of the most important aspects of delivering treatment is learning which treatment is best to provide a user in a given state. Intuitively, providing too few mobile notifications may defeat the purpose of the treatment, while bombarding with too many notifications may increase a sense of burden on the user and reduce the effectiveness of the treatment. Reinforcement learning (RL) can be used to learn policies for delivering treatment that balance this trade-off for each user. However, the application domain of mobile health presents a number of particular challenges. The biggest challenge is the *importance of efficient learning*. Because the nature of reinforcement learning algorithms requires the agent to interact frequently with the environment, the very act of collecting data can cause users to lose interest, leading to disengagement and diminishing the efficacy of the intervention. Another feature of mobile health is the need for *online learning as a continuing task*. Unlike many problems for which training is done offline or in episodes, restarting is not an option in mobile health.

There is a growing body of work which uses RL to learn treatment policies for mobile health. The majority of these have either applied bandit-type algorithms that optimize only for immediate reward [1, 2], or modelled future reward with little focus on efficient exploration [3, 4]. This is problematic since greedy, bandit-type algorithms do not balance the reward obtained vs. the burden incurred by treatment. In the reinforcement learning literature, efficient exploration has been an important topic of study; however, much of this has been in the context of episodic learning [5, 6], which is less applicable to many problems in mobile health that occur in a continuing task setting. Relevant to our work are methods based on posterior sampling, in particular RLSVI [7], which have been shown to explore efficiently across a large class of finite-horizon problems. RLSVI performs posterior sampling of value functions in a least-squares estimation framework, and was shown to be provably more efficient in tabular learning than epsilon-greedy or Boltzmann exploration [7].

This paper studies the applicability of the finite-horizon RLSVI algorithm from [7] to mobile health as a method of efficient exploration, and introduces an extension of the algorithm from its original episodic setting to a continuing task setting. We evaluate the original, finite-horizon RLSVI algorithm using a simulation testbed that reflects a simplified mobile health setting, and we also propose and evaluate a continuing task extension of RLSVI that is more representative of mobile health problems in the real world. We confirm prior findings that RLSVI does indeed outperform LSVI in the episodic, finite-horizon case and also establish RLSVI as performing more robustly than LSVI over a wide range of hyperparameters, in both the episodic and continuing task settings for mobile health.

2 Methods

The goal of this work is to develop an online algorithm that can efficiently learn a policy for determining optimal times to deliver treatment in mobile health. The policy must avoid incurring high burden on the user due to the nature of mobile health problems described in Sec. 1. We define the notion of two types of burden: short-term and long-term. Short-term burden is caused by a single notification or treatment that affects the user temporarily, while long-term burden is caused by the total treatment that the user has undergone which causes them to be less sensitive to future treatments. We define the *burden size* to be a measure of the user's sensitivity to treatment, which corresponds to the quantity of short-term burden accumulated upon receiving a notification.

In this section, we present an approach for efficiently learning treatment policies in mobile health that balance achieving reward (e.g. desired behavior) vs. incurring too much burden. We investigate two settings: the finite-horizon setting and the continuing task setting. First, we describe how we apply the finite-horizon RLSVI algorithm of [7] to an episodic testbed that reflects a simplified mobile health setting. We then propose an extension of RLSVI to the continuing task setting, using a testbed that models the typical non-restarting nature of mobile health problems.

2.1 Finite horizon

While treatment delivery in mobile health is more appropriately modelled as a continuing task in most situations, we can model the problem using a finite horizon when it is reasonable to assume that there are no considerable long term effects. We can imagine a simple scenario in which the user follows a daily "routine" which resets every day, such that learning can occur episodically at the end of each day. Under the finite-horizon assumption, the RLSVI algorithm proposed in [7] can be used. Here, we describe our problem setup in the finite-horizon setting, and our use of RLSVI to enable more efficient learning.

Problem setup. Following the intuition described above, let H denote the finite horizon length corresponding to the episodic time before a reset in the mobile health problem. We model the problem using a two-dimensional state capturing a user's level of short-term and long-term burden, respectively. Specifically, we define the state at timestep t of episode l to be $(s_{l,h}^{(0)}, s_{l,h}^{(1)}) \in \{0, \dots, Hd\} \times \{0, \dots, H\}$. The initial state resets to $(0, 0)$ at the beginning each episode. At each timestep of each episode, an action $a_{l,h} \in \{0, 1\}$ is selected, where 1 corresponds to pinging the user with a mobile

notification and 0 corresponds to waiting (no ping). We define a deterministic transition function $T(s,a) = s'$ such that $(s^{(0)}_{l,h+1}, s^{(1)}_{l,h+1}) = T((s^{(0)}_{l,h}, s^{(1)}_{l,h}), 1) = (s^{(0)}_{l,h} + d, s^{(1)}_{l,h} + 1)$ when a ping action is taken and $(s^{(0)}_{l,h+1}, s^{(1)}_{l,h+1}) = T((s^{(0)}_{l,h}, s^{(1)}_{l,h}), 0) = (\max(0, s^{(0)}_{l,h} - 1), s^{(1)}_{l,h})$ when a wait action is taken. Intuitively, pinging has an increasing effect on both short and long-term burden, while waiting allows the user to recover from the burden. We also define a reward function $R((s^{(0)}, s^{(1)}), a)$ such that $R((s^{(0)}, s^{(1)}), 0) = 0$ given a wait action and $R((s^{(0)}, s^{(1)}), 1) = r_0^{-s^{(0)}} r_1^{-s^{(1)}}$ given a ping action, for fixed $r_0 \geq 1, r_1 \geq 1$. The reward observed from taking action $a_{l,h}$ in state $(s^{(0)}_{l,h}, s^{(1)}_{l,h})$ is then $r_{l,h} = R((s^{(0)}_{l,h}, s^{(1)}_{l,h}), a) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2_{true})$ is random noise with true standard deviation σ_{true}.

RLSVI. In order to learn treatment policies for this mobile health problem, we use the RLSVI algorithm introduced in [7] for the finite-horizon setting. RLSVI has been shown to enable more efficient learning through randomized sampling of value functions (as opposed to actions), in a least-squares value iteration framework. More specifically, the RLSVI algorithm uses a matrix of basis functions Φ to represent the state-action space. It then performs Bayesian linear regression to approximate the value function at each timestep of the finite horizon, starting with the last timestep: $h = H-1, ..., 1, 0$. At each timestep, observed data from previous episodes for that particular timestep is used to perform the linear regression, and the resulting parameter mean and covariance matrix is used to sample a value function that will be used in determining the action at that timestep in the next episode.

Algorithm 1: Continuing task RLSVI

Input: Features $\Phi(s_i, a_i), r_i : i < t$, $\lambda > 0$, $\sigma > 0$, $H > 0$
Output: $\tilde{\theta}_{t,0}, ...\tilde{\theta}_{t,H-1}$
for $h = H-1, ..., 1, 0$ **do**
  Generate regression problem $A \in \mathbb{R}^{(t-H+1)\times K}, b \in \mathbb{R}^{t-H+1}$:
  for $i = 0, 1, ..., t-H$ **do**
    $A_i \leftarrow \Phi(s_{h+i}, a_{h+i})$
    $b_i \leftarrow \begin{cases} r_{h+i} + \max_\alpha \left(\Phi\tilde{\theta}_{t,h+1}\right)(s_{h+i+1}, \alpha) & \text{if } h < H-1 \\ r_{h+i} & \text{if } h = H-1 \end{cases}$
  end
  Bayesian linear regression for the value function
  $$\bar{\theta}_{t,h} \leftarrow \sigma^{-2}\left(\sigma^{-2}A^TA + \lambda I\right)^{-1} A^T b$$
  $$\Sigma_{t,h} \leftarrow \left(\sigma^{-2}A^TA + \lambda I\right)^{-1}$$
  Sample $\tilde{\theta}_{t,h} \sim \mathcal{N}(\bar{\theta}_{t,h}, \Sigma_{t,h})$ from Gaussian posterior
end

2.2 Continuing task

While the finite-horizon testbed is a good starting point for verifying the efficacy of RLSVI, it is not directly applicable to mobile health problems where learning must be continuous and online. Although temporal factors and user routines still exist, it is naive to assume that the user state does a complete "reset" every day. For our continuing task simulations, we use a similar setup to the finite-horizon setting, except that training is continual rather than episodic, and for a given state, the level of long-term burden is determined by the number of pings in a fixed memory window of length T of previous actions, rather than all previous actions (due to the infinite nature of the problem). Instead of the time horizon, we let H be the lookahead horizon on which to perform value iteration, then use the value function corresponding to the first step of the lookahead horizon to determine the next action.

Algorithm 2: Policy learning procedure for continuing tasks

Input: Features Φ; $\sigma > 0, \lambda > 0, H$
for $t = 0, 1, 2, ...$ **do**
  if $t \leq H-1$ **then**
    Use random $\tilde{\theta}_{t,0}, \tilde{\theta}_{t,1}, ..., \tilde{\theta}_{t,H-1} \sim \mathcal{N}(0, 1/\lambda)$
  else
    Obtain $\tilde{\theta}_{t,0}, ..., \tilde{\theta}_{t,H-1}$ from Algorithm 1
  end
  Sample $a_t \in \arg\max_{\alpha \in \mathcal{A}} \left(\Phi\tilde{\theta}_{t,0}\right)(s_t, \alpha)$
  Observe r_t and s_{t+1}
end

Problem setup. To model the continuing task, we now no longer have an episode length or restarts to the initial state at the beginning of each episode. Instead, we define a memory window T, which is the time window over which long-term burden can accumulate. Our two-dimensional state at each timestep t is now $(s^{(0)}_t, s^{(1)}_t) \in \{0, \ldots, d(T+1)\} \times \{0, \ldots, T\}$. At each timestep, an action $a_t \in \{0, 1\}$ is selected. As in the finite-horizon setting, 1 corresponds to pinging the user with a mobile notification and 0 corresponds to waiting (no ping). Our deterministic transition function is $T_{inf}(s,a) = s'$ such that $(s^{(0)}_{t+1}, s^{(1)}_{t+1}) = T_{inf}((s^{(0)}_t, s^{(1)}_t), 1) = (\min(s^0_t + d, d(T+1)), p_t)$ when a ping action is taken and $(s^{(0)}_{t+1}, s^{(1)}_{t+1}) = T_{inf}((s^{(0)}_t, s^{(1)}_t), 1) = (\max(0, s^{(0)}_t - 1), p_t)$ when a wait action is taken, where $p_t = \sum_{i=\max(0,t-T+1)}^{t} 1_{a_i=1}$ is the number of pings in the memory window of the last T timesteps (including action a_t). We use the same reward function $R((s^{(0)}, s^{(1)}), a)$ defined in the finite-horizon problem such that for a given state $(s^{(0)}_t, s^{(1)}_t)$, the reward observed from taking action a_t is $r_t = R((s^{(0)}_t, s^{(1)}_t), a_t) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2_{true})$ is random noise.

Extending RLSVI for use with continuing tasks. In order to use RLSVI for the continuing task setting that is prevalent in mobile health, we extend it from finite to infinite horizon as presented in Algorithm 1. The full learning algorithm for continuing tasks is presented in Algorithm 2. Our extension still retains roughly the same structure as the RLSVI algo-

rithm in [7]; however, we make two important modifications: since the problem is no longer episodic, rolling lookaheads of length H are used in place of the length H episodes in [7], and the value functions approximate the expected total reward over the next H time steps for each state. For initial timesteps $t = 0, ..., H-1$, random actions are taken to accumulate observations. Then as described in Algorithm 2, at every subsequent timestep $t = H$ and onwards, Algorithm 1 is used to output H policies $\tilde{\theta}_{t,h}$ for $h = H-1, ..., 0$, corresponding to each step of the length-H lookahead. The policy $\tilde{\theta}_{t,h}$ is obtained through Bayesian linear regression as in the finite-horizon case; however, due to the lack of episodes, the parameter mean and covariance matrix $\bar{\theta}_{t,h}, \Sigma_{t,h}$ are computed using observations from the hth step of all previously observed stretches of length-H consecutive timesteps. At timestep t, there are a total of $t - H$ previously observed such length-H "episodes" i, so row A_i of matrix A contains the observed feature vector for timestep $i + h$, while element b_i of regression target vector b is the expected total reward over the next H steps. When $h = H - 1$, regression target b_i is determined using the policy $\tilde{\theta}_{t-1,H-1}$ obtained from RLSVI at the previous timestep $t-1$; for all other h, b_i is determined using the policy $\tilde{\theta}_{t,h+1}$. While this procedure corresponds to computing policies looking H timesteps into the future, the next action a_t is always selected using only the policy $\tilde{\theta}_{t,0}$ corresponding to the first timestep of the lookahead. To summarize, we solve the continuing task problem by formulating it as a series of finite-horizon problems which use rolling lookaheads of length H. Our formulation is motivated in spirit by algorithms in model predictive control which perform iterative, finite-horizon planning over short time scales in complex systems.

3 Experiments

In this section, we present experiments evaluating RLSVI in the finite-horizon and continuing task settings for mobile health, using simulation testbeds built to model these settings.

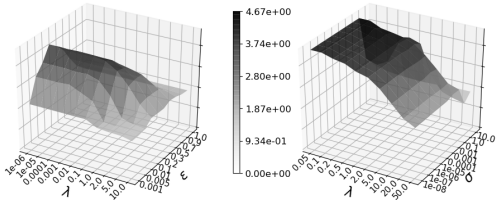
Baseline. We compare with Least-Squares Value Iteration (LSVI) with ϵ-greedy exploration as our baseline in these experiments. LSVI has the same framework as RLSVI in that it performs linear regression to compute the approximation of expected future reward at each timestep using observed data from previous episodes. However rather than sampling from a posterior distribution, LSVI uses the mean obtained from the linear regression as the estimate for the expected future reward, and thus has no built-in exploration. Therefore ϵ-greedy must be used in addition to LSVI to encourage exploration. We experiment with different ϵ values in our simulations. See [7] for a juxtaposition of LSVI and RLSVI.

Implementation details. For all simulations, we use one-hot vector representations of the state-action space as our feature matrix Φ and reward standard deviation $\sigma_{true} = 0.01$. For the initial episodic finite-horizon simulations, we used a time horizon of $H = 16$ for burden size $d = 3$ and reward decay rates $r_0 = 1.5, r_1 = 1.0$ for the reward function, modeling a treatment with no long-term burden effects. Finite-horizon simulations were run for 1000 episodes of 16 timesteps each. In the continuing task simulations, we consider burden size $d = 3$, memory window $T = 4$, lookahead horizon $H = 8$, and reward decay rates $r_0 = 1.5, r_1 = 1.2$ for the reward function, modeling a treatment where long-term burden effects are now present. We run all continuing task experiments for 10000 timesteps and used discount factor $\gamma = 0.95$. For all experiments, a fixed random seed was used over all hyperparameters for a given trial to allow for comparability.

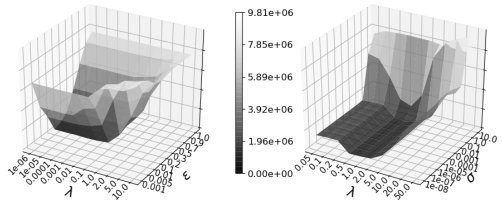
Evaluation. To compare RLSVI and LSVI, we evaluate performance over a broad range of hyperparameters using two metrics: the final reward obtained after the end of the learning process, and the mean-squared error of the total reward obtained by the learned policy as compared to the optimal policy. The final reward is calculated by taking the average total reward obtained in the last 100 episodes (in the finite-horizon case) or last 100 timesteps (in the continuing task case). The MSE is computed as $(\overline{r} - r_{opt})^2 + \frac{1}{N}\sum_{n=1}^{N}(r_n - \overline{r})^2$, where N is the number of trials that were performed, r_n is the total reward obtained in the nth trial, $\overline{r}$ is the mean total reward averaged over N trials, and r_{opt} is the maximum reward achievable using the optimal policy.

3.1 Finite horizon

We first analyze our results for the finite-horizon setting. As discussed in Sec. 2.1, this is a simpler setting corresponding to problems in mobile health with little to no long term effects, and also provides useful intuition for comparing RLSVI and LSVI. Using our testbed for this setting, the optimal policy achieves a final reward of 4.67. Considering the MSE metric, RLSVI achieves a lowest MSE of $3.53 \cdot 10^5$ and final reward 4.25 with hyperparameters $(\lambda, \sigma) = (1, 0.1)$, while LSVI achieves a lowest MSE of $3.91 \cdot 10^5$ and final reward 4.28 with $(\lambda, \epsilon) = (10^{-5}, 0.1)$. Considering the highest final reward metric, RLSVI attains the optimal policy reward of 4.67 with MSE $1.48 \cdot 10^6$ using $(\lambda, \sigma) = (0.1, 0.01)$, while LSVI is not able to learn the optimal policy reward and only obtains final reward 4.28 with MSE $3.91 \cdot 10^5$ using $(\lambda, \epsilon) = (10^{-5}, 0.1)$. For the hyperparameters that achieve low MSE, RLSVI sacrifices some exploration to achieve the low MSE, and therefore RLSVI and LSVI both achieve similar MSE and final reward. However, for the hyperparameters that achieve high final reward, RLSVI has higher MSE because the initial 500 episodes are a period of high exploration and lower rewards, which allows the algorithm to learn the optimal policy while LSVI does not. From Figure 1, we can also see that overall, RLSVI performance is more robust over different parameter values compared to LSVI, which seems to perform poorly

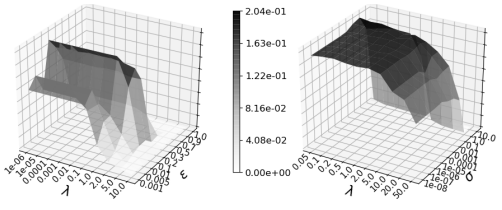


(a) Final reward achieved for LSVI (left) vs. RLSVI (right)

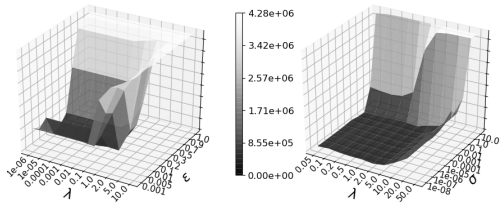


(b) MSE for LSVI (left) vs. RLSVI (right)

Figure 1: Finite-horizon simulation results (burden size $d = 3$, horizon $H = 16$)



(a) Final reward achieved for LSVI (left) vs. RLSVI (right)



(b) MSE for LSVI (left) vs. RLSVI (right)

Figure 2: Continuing task simulation results (burden size $d = 3$, lookahead $H = 8$, window $T = 4$)

for epsilon values besides 0.1. This is significant in mobile health as the optimal hyperparameters are not known before clinical trials begin, and thus it is important for algorithms to be robust and effective across a large range of parameters.

3.2 Continuing tasks

In the continuing task setting, we observe similar behavior in comparing RLSVI vs. LSVI. For reference, the average final reward obtained by the optimal continuing task policy is 0.2083. Considering the MSE metric, RLSVI achieves a lowest MSE of $9.35 \cdot 10^4$ and final reward 0.1915 with $(\lambda, \sigma) = (2, 10^{-4})$, while LSVI achieves a lowest MSE of $4.71 \cdot 10^4$ and final reward 0.1947 with $(\lambda, \epsilon) = (10^{-3}, 0.1)$. Considering the highest final reward metric, RLSVI attains final reward 0.2041, almost the optimal reward, with MSE $1.18 \cdot 10^5$ using $(\lambda, \sigma) = (0.05, 0.1)$, while LSVI does not quite achieve optimal final reward and attains 0.1951 with MSE $9.62 \cdot 10^4$ using $(\lambda, \epsilon) = (1, 0.1)$. Due to the explorative nature of RLSVI, the MSE of RLSVI tends to be higher than LSVI, but we observe that Figure 2 again illustrates the improved robustness of RLSVI over different hyperparameters, compared to LSVI.

4 Conclusion

In this paper, we both investigated the application of finite-horizon RLSVI to mobile health, and introduced an extension of RLSVI for continuing tasks that is able to learn continuously rather than episodically and therefore model the more prevalent and complex continuing task setting in mobile health. In simulation experiments for these mobile health problems, we showed that RLSVI outperforms LSVI and is additionally more robust than LSVI on both finite and continuing task problems over various hyperparameters. We intend to continue investigating the extended RLSVI algorithm for continuing tasks on increasingly larger state spaces and testing it using non-stationary rewards and higher noise environments, with the aim of applying it to a continuing task mobile health problem in an ongoing real-world study.

References

[1] Mashfiqui Rabbi, Min Hane Aung, Mi Zhang, and Tanzeem Choudhury. Mybehavior: automatic personalized health feedback from user behaviors and preferences using smartphones. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 707–718. ACM, 2015.

[2] Elad Yom-Tov, Guy Feraru, Mark Kozdoba, Shie Mannor, Moshe Tennenholtz, and Irit Hochberg. Encouraging physical activity in patients with diabetes: intervention using a reinforcement learning system. *Journal of medical Internet research*, 19(10), 2017.

[3] Mo Zhou, Yonatan Mintz, Yoshimi Fukuoka, Ken Goldberg, Elena Flowers, Philip Kaminsky, Alejandro Castillejo, and Anil Aswani. Personalizing mobile fitness apps using reinforcement learning. In *IUI Workshops*, 2018.

[4] Shanice Clarke, Luis G Jaimes, and Miguel A Labrador. mstress: A mobile recommender system for just-in-time interventions for stress. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–5. IEEE, 2017.

[5] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[6] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[7] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. 2016.

A cognitive tutor for helping people overcome present bias

Falk Lieder[*]
Max Planck Institute for Intelligent Systems
Tübingen, Germany
falk.lieder@tuebingen.mpg.de

Frederick Callaway[*]
Princeton University
Princeton, NJ, USA
fredcallaway@princeton.edu

Yash Raj Jain
Max Planck Institute for Intelligent Systems
Tübingen, Germany
f20140604@hyderabad.bits-pilani.ac.in

Paul M. Krueger
Princeton University
Princeton, NJ, USA
paul.m.krueger@gmail.com

Priyam Das
UC Irvine
Irvine, CA, USA
priyam.das@uci.edu

Sayan Gul
UC Berkeley
Berkeley, CA, USA

Thomas L. Griffiths
Princeton University
Princeton, NJ, USA
tomg@princeton.edu

[*] These authors contributed equally.

Abstract

People's reliance on suboptimal heuristics gives rise to a plethora of cognitive biases in decision-making including the *present bias*, which denotes people's tendency to be overly swayed by an action's immediate costs/benefits rather than its more important long-term consequences. One approach to helping people overcome such biases is to teach them better decision strategies. But which strategies should we teach them? And how can we teach them effectively? Here, we leverage an automatic method for discovering rational heuristics and insights into how people acquire cognitive skills to develop an intelligent tutor that teaches people how to make better decisions. As a proof of concept, we derive the optimal planning strategy for a simple model of situations where people fall prey to the present bias. Our cognitive tutor teaches people this optimal planning strategy by giving them metacognitive feedback on how they plan in a 3-step sequential decision-making task. Our tutor's feedback is designed to maximally accelerate people's metacognitive reinforcement learning towards the optimal planning strategy. A series of four experiments confirmed that training with the cognitive tutor significantly reduced present bias and improved people's decision-making competency: Experiment 1 demonstrated that the cognitive tutor's feedback can help participants discover far-sighted planning strategies. Experiment 2 found that this training effect transfers to more complex environments. Experiment 3 found that these transfer effects are retained for at least 24 hours after the training. Finally, Experiment 4 found that practicing with the cognitive tutor can have additional benefits over being told the strategy in words. The results suggest that promoting metacognitive reinforcement learning with optimal feedback is a promising approach to improving the human mind.

Keywords: cognitive training; planning; metacognitive reinforcement learning; cognitive plasticity

Acknowledgements

This work was supported by grant number ONR MURI N00014-13-1-0341 and a grant from the Templeton World Charity Foundation to TLG. We thank Tania Lombrozo, Peter Dayan, Thomas Hills, and Mike Mozer for helpful comments and discussions.

1 Introduction

Research on heuristics and biases has identified many ways in which human judgment and decision-making might be sub-optimal (Tversky and Kahneman, 1974). For instance, decision-makers often partly or entirely neglect the long-term consequences of their choices while being overly swayed by their immediate costs or benefits in the present. This phenomenon is known as the *present bias* (O'Donoghue and Rabin, 1999) and gives rise to procrastination, impulsivity, and poor economic decisions, such as people's failure to save for retirement. Interventions that help people overcome present bias could thus confer significant benefits to individuals, organizations, and society.

Two of the main challenges for improving the human mind are a) discovering effective cognitive strategies, and b) teaching them effectively so that people will apply them in everyday life. Our recent work has begun to address the first problem by developing automatic methods for deriving resource-rational cognitive strategies (Lieder and Griffiths, 2019; Lieder et al., 2017). Here, we address the second problem by developing and evaluating a cognitive tutor that teaches people far-sighted planning strategies.

We start by introducing the experimental paradigm we use to observe and intervene on the decision strategies that might give rise to the present bias. We then derive a resource-rational planning strategy for overcoming the present bias. Next, we present a cognitive tutor that teaches people this cognitive strategy via a metacognitive feedback mechanism. Finally, we evaluate our cognitive tutor in a series of experiments and discuss our findings and directions for future work.

2 Measuring present bias with the Mouselab-MDP paradigm

Previous work has inferred present bias from the choices that people make. But the underlying mechanisms remain to be elucidated. Here, we use a recently developed process-tracing paradigm (Callaway et al., 2017) to measure the decision strategies that might give rise to present bias. Making future-minded decisions requires planning. Planning, like all cognitive processes, cannot be observed directly but has to be inferred from observable behavior. To be able to give people feedback on the quality of their planning strategies we therefore draw on the Mouselab-MDP paradigm (Callaway et al., 2017) illustrated in Figure 1a and 1c to make people's behavior more diagnostic of their planning strategies. This paradigm externalizes people's beliefs and planning operations as observable states and actions (Callaway et al., 2018,1). Inspired by the Mouselab paradigm (Payne et al., 1993), it uses people's mouse-clicking as a window into their planning. Participants are presented a series of route planning problems where each location (the gray circles) harbors a gain or loss. These potential gains and losses are initially occluded, corresponding to a highly uncertain belief state. The participant can reveal each location's reward by clicking on it and paying a fee of \$1.

The key property of situations in which present bias impairs human decision-making is the misalignment between immediate reward and long-term value. As an illustration of this problem, consider the choice between beginning work on a manuscript and watching a YouTube video. Staring at a blank page might make you feel anxious in the short run, but you will feel very satisfied when you submit the paper for publication months later. By contrast, the YouTube video will give you immediate joy but you might later come to regret the wasted time. To make good decisions in situations like this, people have to look beyond the salient immediate rewards, set a goal for the future, plan how to achieve it, and execute the plan. What makes this far-sighted approach worthwhile is that the range of outcomes that can be obtained by a concerted sequence of actions over an extended period of time is much larger than the range of rewards that can be attained immediately.

To capture this aspect of many real-world situations within the Mouselab paradigm, we constructed 3-step sequential decision environments where the range of rewards increases from the first step to the second step and was largest in the third step. In each episode, rewards are independently drawn from discrete uniform distributions; in the first step the possible values were $\{-4, -2, +2, +4\}$; in the second step the possible values were $\{-8, -4, +4, +8\}$; and in the third step the possible values were $\{-48, -24, +24, +48\}$. To plan their actions participants can uncover the rewards at each location by clicking on it for a fee of \$1 per click. This captures that the decision-maker's time is costly.

Recording the clicks people make in this paradigm allows us to detect whether their decisions are swayed by present bias. That is, if a participant only inspects the immediate outcomes of the first step while ignoring the outcomes of the second step and the third step, then we know that their decision was affected by present bias. Conversely, if a participant looks at the potential final outcomes in the third step while ignoring the immediate outcomes, we can be confident that they were not swayed by present bias. Our paradigm thereby allows us to i) observe the maladaptive heuristics that give rise to present bias, and to ii) trace whether and how they improve in response to interventions. To develop an effective intervention, the next section derives the optimal planning strategy for the environment modelled by this paradigm.

3 Discovering optimal planning strategies that counter present bias

Teaching clever heuristics is a promising approach to improving decision-making (Gigerenzer and Todd, 1999; Hertwig and Grüne-Yanoff, 2017). But which heuristics should be taught and how can we discover such heuristics? The theory of *resource-rationality* provides a mathematically precise definition of optimal heuristics (Lieder and Griffiths, 2019). In essence, the optimal heuristic for a decision-maker to use in a given environment achieves the best possible tradeoff between the expected utility of the resulting decision

and the expected opportunity cost of its execution. In the Mouselab-MDP paradigm, heuristics can be expressed as rules for deciding where to click given which information has already been revealed, when to stop clicking, and where to move given the information that has been uncovered. In previous work, we derived the optimal planning strategies for several Mouselab-MDP environments by solving metalevel MDPs using backward induction (Callaway et al., 2018).

In particular, Callaway et al. (2018) found that the resource-rational planning strategy for the environment described in Section 2 is to first set a goal by evaluating potential final destinations. As soon as inspecting a potential final destination uncovers the highest possible reward (+48), the optimal strategy selects the path that leads to it and terminates planning. If all potential final destinations have been inspected and one was revealed to be better than all the others, then the optimal strategy immediately decides to go there; else the optimal strategy inspects additional nodes located immediately before those most promising final destination and then chooses the path that is most promising according to the revealed information (and stops planning as soon as one path is revealed to be as good as possible). Having discovered this optimal planning strategy, we now present a cognitive tutor that teaches it to people.

4 Countering present bias with cognitive tutoring

We developed a cognitive tutor that teaches cognitive strategies by giving people metacognitive feedback. Our tutor's pedagogy is based on findings suggesting that people learn how to decide at least partly from the rewards and punishments they experience as a consequence of their decisions (Krueger et al., 2017; Lieder and Griffiths, 2017). This evidence for *metacognitive reinforcement learning* suggests that it should be possible to apply methods that have been developed to accelerate model-free reinforcement learning in robots– such as reward shaping (Ng et al., 1999)– to accelerate metacognitive learning in people. Following this line of reasoning, we used the following reward shaping method to generate optimal feedback signals for accelerating the process by which people learn how to make better decisions:

1. Model the cognitive function to be improved (i.e., planning) and the available cognitive operations (e.g., simulating the outcome of taking a certain action in a certain state) and their costs as a metalevel Markov Decision Process (MDP).
2. Compute the values of the computations c people might perform in different belief states b (i.e., the state-action value function $Q_{\text{meta}}(b,c)$) by solving the metalevel MDP.
3. Let people practice the cognitive function to be improved and infer their computations from process tracing data.
4. Score people's inferred computations by

$$\text{score}(b,c) = \hat{Q}_{\text{meta}}(b,c) - \max_c \hat{Q}_{\text{meta}}(b,c). \tag{1}$$

5. Translate score into reinforcement and a feedback message.

We completed Step 1 and Step 2 in previous work (Callaway et al., 2018). Step 3 is accomplished by using the Mouselab-MDP paradigm to measure people's planning operations. Finally, the feedback signal computed in Step 4 is translated into a delay penalty of $2 - \text{score}$ seconds if the participant made an error or 0 seconds if their planning operation was optimal. The resulting feedback is given immediately after each click.

The cognitive tutor shown in Figure 1a integrates this feedback mechanism into the Mouselab-MDP paradigm and augments it with demonstrations of the optimal strategy described in Section 3. That is, the tutor's feedback has two components: i) a delay penalty whose duration communicates how sub-optimal the participant's planning operation was, and ii) a hint about what the optimal strategy would have done differently. Concretely, if the tutee makes an error then the planning operation that the optimal strategy would have performed instead is highlighted in blue (see Figure 1a). By contrast, when participants respond correctly, then they are told that they did a good job and can move on to the next click immediately.

5 Results

We evaluated our cognitive tutor in four online experiments that were run on Amazon Mechanical Turk. For each of these experiments we recruited about 50 participants per condition. Participants were paid performance-based bonuses.

In the control condition of each experiment, participants solved 31 different 3-step sequential decision problems in a plain version of the Mousleab-MDP paradigm shown in Figure 1a. Inspecting the recorded click sequences revealed that 38% of the participants in the control condition exhibited the present bias on the first trial. We identified three distinct planning strategies that gave rise to the present bias: i) a myopic satisficing strategy that inspects the immediate outcomes of alternative actions until it encounters a positive outcome and then immediately chooses the corresponding action, ii) a myopic maximizing strategy that inspects each action's immediate outcomes and then chooses the action with the best immediate outcome, and iii) an overly frugal myopic strategy that inspects only a single immediate outcome and nothing else.

Experiments 1-3 employed a between-subjects pre-post design comparing the effects of practicing the task with versus without the cognitive tutor. In Experiment 1, the pre-test, training, and post-test blocks all employed the same 3-step planning task shown in

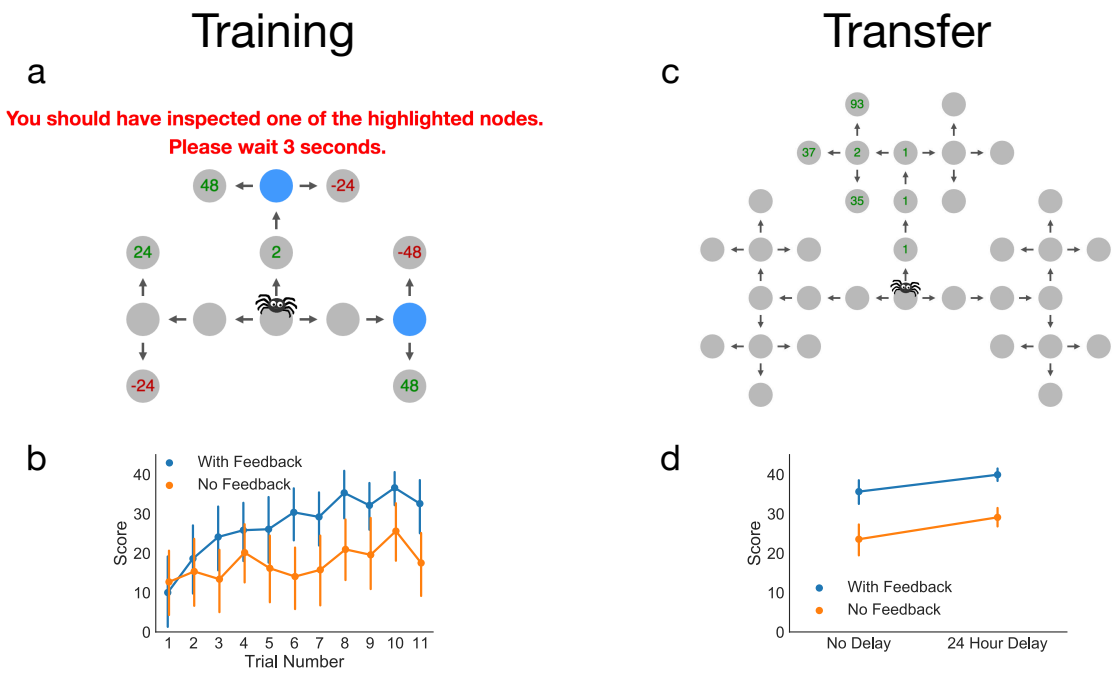


Figure 1: (a) Example feedback from the cognitive tutor in the training phase. (b) Participants learn to achieve high scores faster with the tutor's feedback. (c) A more difficult transfer problem. Feedback is not given in either condition. (d) Participants who received feedback in the training phase outperform control participants when tested immediately or after a 24 hour delay.

Figure 1a. We found that the tutor's metacognitive feedback significantly improved our participants' learning (see Figure 1b) and led to significantly higher post-test performance (36.2 \$/trial vs. 24.6 \$/trial, $t(2258) = 10.7$, $p < 0.0001$).

To elucidate how these improvements in performance were accomplished, we inspected how the prevalence of different types of strategies changed over time in the presence versus absence of optimal metacognitive feedback. We found that, over time, participants in the control condition slowly developed more adaptive planning strategies. In this process the prevalence of near-optimal goal-setting strategies increased from only 0.0% in the first trial to 26.4% after 30 trials ($\chi^2(1) = 16.1$, $p < .0001$). Conversely, the prevalence of the sub-optimal strategies giving rise to present bias dropped from 37.7% to 9.4% ($\chi^2(1) = 11.8$, $p = .0006$) and the prevalence of acting impulsively without any planning decreased from 33.9% to 26.4% ($\chi^2(1) = 0.72$, $p = .3974$). As shown in Figure 2, our tutor's optimal feedback amplified both of these changes, thereby increasing the the prevalence of near-optimal goal setting from 0.0% to 50.9% ($\chi^2(1) = 34.9$, $p < .0001$) while decreasing the prevalence of short-sighted decision strategies from 45.1% to 0.0% ($\chi^2(1) = 29.7$, $p < .0001$) and the prevalence of impulsive choices from 29.4% to 5.9% ($\chi^2(1) = 9.7$, $p = .0018$). Critically, we found that the feedback of our cognitive tutor significantly increased the proportion of people who discovered the far-sighted goal setting strategy from 26.4% to 50.9% ($\chi^2(1) = 6.63$, $p = .0100$) and significantly decreased the eventual prevalence of the maladaptive short-sighted strategies from 9.4% to 0.0% ($\chi^2(1) = 5.05$, $p = .0246$) and the prevalence of the maladaptive impulsive strategy from 26.4% to 5.9% ($\chi^2(1) = 8.01$, $p = .0046$). Furthermore, the learning curves shown in Figure 2 suggest that our tutor's feedback accelerated this transition.

In Experiment 2, the training block used a flight planning task that was structurally equivalent to task used in Experiment 1, whereas the pre-test and post-test blocks used the transfer task shown in Figure 1c). The training and the transfer task were structurally similar in that the variance of the reward distribution increased from each step to the next – being smallest for the immediate rewards and largest for the rewards attainable in the final step. But the transfer task used a different cover story (moving a money loving spider through a web of cash vs. flight planning), was more complex (5-step planning vs. 3-step planning) and involved a larger number of possible payoffs (192 vs. 10). As shown in Figure 1d, we found significant transfer effects from the relatively simple 3-step training task to the more difficult 5-step transfer task. Participants who had practiced with the cognitive tutor performed significantly better on the transfer task than participants who had practiced planning without the assistance of our cognitive tutor (37.4 \$/trial vs. 27.4 \$/trial, $t(2358) = 8.8$; $p < .0001$). This transfer effect appears to be partially mediated by people learning to plan backward – which was also beneficial in the transfer task. Concretely, participants who had practiced with the cognitive tutor were more likely to start planning by inspecting one of the final destinations than the control group (91.4% vs. 83.1%, $Z = 3.43$, $p = .0006$) and were less likely to start by inspecting one of the rewards in the first step (2.21% vs. 14.8%, $Z = -6.33$, $p < 0.0001$). In Experiment 3, we found that the transfer effect was after a delay of approximately 24 hours (39.9 \$/trial vs. 39.1 \$/trial, $t(1578) = 7.8$; $p < .0001$).

Experiment 4 compared the effectiveness of instruction plus practice with the cognitive tutor versus pure instruction and instruction plus watching a video demonstration of the optimal strategy. Participants in all three conditions read about the goal-setting principle for better decision-making discovered by Callaway et al. (2018). In the experimental conditions participants subsequently practiced applying the goal-setting principle with the cognitive tutor or saw a video demonstration of the optimal strategy. After 24 hours all participants were tested on the transfer task (Figure 1c). We found that participants who had practiced with the cognitive tutor performed significantly better on the transfer task than participants who were only told about the principle (38.0 \$/trial vs. 24.2 \$/trial, $t(83) = 10.5, p = 0.0000$). Participants who had seen a demonstration of the optimal strategy performed at the same level as participants who had practiced with the cognitive tutor (38.8 \$/trial vs. 38.0 \$/trial, $t(78) = -0.7, p = 0.49$). In either case, our cognitive tutor significantly improved people's planning by teaching them the resource-rational strategy we derived in previous work.

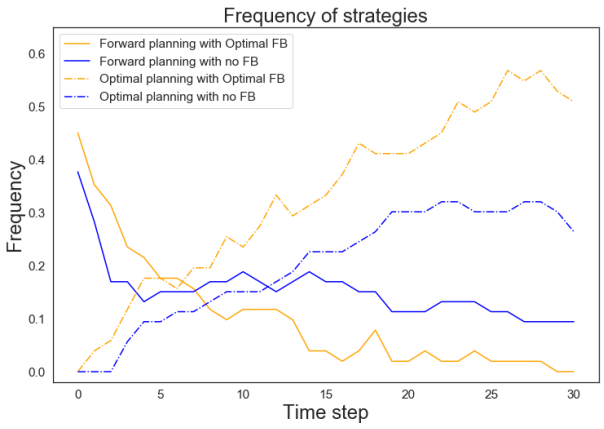


Figure 2: Prevalence of short-sighted versus far-sighted planning strategies for participants practicing with versus without out cognitive tutor.

6 Discussion

The present work illustrates how artificial intelligence can be leveraged to discover and teach rational decision-making strategies. The theoretical framework of resource-rationality allowed us to derive near-optimal planning strategies automatically, and the theory of metacognitive reinforcement learning allowed us to develop an intelligent system that can teach those rational heuristics very effectively. Our preliminary results suggest that practice with our cognitive tutor is more effective than instruction and has transferable benefits that are retained over time. Concretely, we found that participants who practiced decision-making with our cognitive tutor were significantly more likely to overcome the present bias by learning more far-sighted decision strategies than participants who practiced the same task without the tutor. This suggests that combining automatic strategy discovery with intelligent tutors is a promising approach to enhancing human rationality.

Moving forward we will investigate how diagnostic our paradigm is of people's propensity to succumb to the present bias in everyday life and whether our approach to enhancing human rationality can be used to improve the decisions that people make in the real world. In future work we will also apply our approach to helping people overcome additional cognitive biases and training them to put more thought into important decisions.

References

Callaway, F., Lieder, F., Das, P., Gul, S., Krueger, P. M., and Griffiths, T. L. (2018). A resource-rational analysis of human planning. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, Austin, TX. Cognitive Science Society.

Callaway, F., Lieder, F., Krueger, P. M., and Griffiths, T. L. (2017). Mouselab-MDP: A new paradigm for tracing how people plan. In *The 3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making, Ann Arbor, MI*.

Gigerenzer, G. and Todd, P. M. (1999). *Simple heuristics that make us smart*. Oxford University Press, New York, NY.

Hertwig, R. and Grüne-Yanoff, T. (2017). Nudging and boosting: Steering or empowering good decisions. *Perspectives on Psychological Science*, 12(6):973–986.

Krueger, P. M., Lieder, F., and Griffiths, T. L. (2017). Enhancing metacognitive reinforcement learning using reward structures and feedback. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society*. Cognitive Science Society.

Lieder, F. and Griffiths, T. (2017). Strategy selection as rational metareasoning. *Psychological Review*, 124:762 –794.

Lieder, F. and Griffiths, T. L. (2019). Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*.

Lieder, F., Krueger, P. M., and Griffiths, T. L. (2017). An automatic method for discovering rational heuristics for risky choice. In Gunzelmann, G., Howes, A., Tenbrink, T., and Davelaar, E. J., editors, *Proceedings of the 39th Annual Meeting of the Cognitive Science Society*, pages 742–747, Austin, TX. Cognitive Science Society.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In Bratko, I. and Dzeroski, S., editors, *Proceedings of the 16th Annual International Conference on Machine Learning*, pages 278–287, San Francisco, CA. Morgan Kaufmann.

O'Donoghue, T. and Rabin, M. (1999). Doing it now or later. *American Economic Review*, 89(1):103–124.

Payne, J. W., Bettman, J. R., and Johnson, E. J. (1993). *The adaptive decision maker*. Cambridge university press.

Tversky, A. and Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157):1124–1131.

Investigating Curiosity for Multi-Prediction Learning

Cameron Linke
University of Alberta
clinke@ualberta.ca

Nadia M. Ady
University of Alberta
nmady@ualberta.ca

Thomas Degris
DeepMind London
degris@google.com

Martha White
University of Alberta
whitem@ualberta.ca

Adam White
University of Alberta
DeepMind Edmonton
adamwhite@google.com

Abstract

This paper investigates a computational analog of curiosity to drive behavior adaption in learning systems with multiple prediction objectives. The primary goal is to learn multiple independent predictions in parallel from data produced by some decision making policy—learning for the sake of learning. We can frame this as a reinforcement learning problem, where a decision maker's objective is to provide training data for each of the prediction learners, with reward based on each learner's progress. Despite the variety of potential rewards—mainly from the literature on curiosity and intrinsic motivation—there has been little systematic investigation into suitable curiosity rewards in a pure exploration setting. In this paper, we formalize this pure exploration problem as a multi-arm bandit, enabling different learning scenarios to be simulated by different types of targets for each arm and enabling careful study of the large suite of potential curiosity rewards. We test 15 different analogs of well-known curiosity reward schemes, and compare their performance across a wide array of prediction problems. This investigation elucidates issues with several curiosity rewards for this pure exploration setting, and highlights a promising direction using a simple curiosity reward based on the use of step-size adapted learners.

Keywords: Reinforcement Learning, Curiosity, Online Learning

Acknowledgements

We are very grateful to the Alberta Machine Intelligence Institute (Amii), the Reinforcement Learning and Artificial Intelligence lab (RLAI), and DeepMind for their support of our work.

1 Introduction

We consider the case of a lifelong learning agent that receives one stream of experience with many different incoming streams of data (e.g. distance sensors, camera, battery state, etc.). This type of agent is outlined in the Horde architecture (Sutton et al., 2011) where a robotic agent learns about many different sensorimotor streams, off-policy, in parallel, while interacting with its environment. This architecture has been shown to scale up to making thousands of predictions at once, giving the agent a rich prediction-based understanding of its environment. Other architectures such as Universal Value Function Approximators (Schaul et al., 2015) and UNREAL (UNsupervised REinforcement and Auxiliary Learning) (Jaderberg et al., 2016) have shown that in complex environments with deep neural networks as function approximators, additional predictions like these are powerful in improving the ultimate performance of the agent. The extra predictions were used to guide the agent to new areas of the state space, or to add richness to a sparse reward signal. Here we consider how the agent's behaviour changes the quality of its predictions of the signals themselves, in contrast to using the predictions as a target to guide behaviour appropriate to a separate task.

For an agent that is able to learn about many things at once, the challenge remains—how does an agent who gets only one stream of experience decide what to learn, and how long to learn it for? Thought of another way—given a number of target policies, what is the best behaviour policy for the agent to learn to predict and control the streams of data that it is receiving? Prior work on this problem has either used a hand-crafted policy designed to explore the environment in a specific way for the learners (Sutton et al., 2011; Modayil et al., 2014), or had the agent target maximizing one specific reward signal and learning about other signals at the same time (Jaderberg et al., 2016; Schaul et al., 2015; Riedmiller et al., 2018). A number of authors have proposed special reward signals to motivate agent learning; herein we call such signals *curiosity rewards*.

While our ultimate goal is to get to a full Horde setting, it remains a complex environment for testing different types of curiosity reward signals to drive agent behavior. The function approximation used by the agent means that we need to consider representation learning in addition to learning to predict the signal itself. The off-policy nature of the signals being learned also complicates the agent as it brings in choices around the type of learners to use, and further need for sweeps over a larger number of parameters. To specifically study the mechanisms driving the curious agent, we want to be able to remove as many of these other complicating factors as possible and focus on the mechanisms themselves.

This paper has three contributions. First we propose a new benchmark for evaluating curiosity-driven learning agents. We propose using a bandit setting to focus on the single learning decision that the agent needs to take at each time step—what action can I take that will maximize learning? This is different from the traditional exploration/exploitation trade-off given by the bandit literature. We are instead focused on maximizing the long-term learning of the agent, not the long-term reward. The setup of our bandit is to have multiple arms each with their own Least Mean Square (LMS) learner predicting the value of the arm. Each arm's learner can be thought of as a single prediction about a sensorimotor stream, and the agent can take an action that will teach it more about that stream of data. This setup models the situation that is faced by an agent in the Horde setting—at each step it is learning about a set of predictions, and a curious agent will want to take an action that allows it to learn the most. The bandit is set up to model the types of challenges a curious agent will face—how does an agent learn to best improve its predictions while avoiding getting stuck on unlearnable signals? Mechanisms that are effective at driving curious behavior in this setting should be effective in the larger setting. Alternatively, mechanisms that are not able to solve this distilled setting will not be effective in the larger Horde setting.

Second, we propose a comprehensive empirical comparison of current approaches to curiosity on our proposed benchmark. We compare many of the well-known methods from reinforcement learning and active learning. Much recent work has been in large-scale domains where it is difficult to tease out which of the many moving parts is driving the benefit of the method. In this comparison we distill the methods down to the specific mechanism driving curious behavior and compare them in our focused domain. This allows us to both compare how the mechanisms perform against each other, as well as evaluate the specific properties of how each mechanism deals with the problems each environment tests. While we did not compare every curiosity method that exists, we did make sure to have a representative method from each of the different classes we identified as grouping similar underlying mechanisms.

Third, we show the effectiveness of capable learners in directing curious behavior. Capable learners, such as step-size adaptation methods, are able to quickly adapt to unlearnable targets, such as those with high variance. This allows the agent to avoid unlearnable signals that may seem interesting due to their high error, but ultimately are not worth the agent continuing to spend time trying to learn. In this work, we show one particular curiosity reward—Weight Change—becomes one of the best curiosity rewards in this setting when paired with capable learners. While we examine capable learners in a simplified setting with LMS learners, nothing precludes these results from being moved back to the Horde setting. The more general case of off-policy learners with function approximation also have step-size adaptation methods (Jacobsen et al., 2019; Kearney et al., 2018) that allow capable learners to be applied.

2 Related Work

Many learning systems draw inspiration from the exploratory behavior of young humans and animals, uncertainty reduction in active learning, and information theory—and the resulting techniques could all be packed into the suitcase of curiosity and intrinsic motivation. In an attempt to distill the key ideas and perform a meaningful yet inclusive empirical study, we consider only methods applicable to our bandit formulation of multi-prediction learning. As such, we ignored work related to curriculum learning (Graves et al., 2017), methods that rely on predictions about state (e.g., (Pathak et al., 2017)), or traditional bandit exploration methods.

Several curiosity rewards are based on **violated expectations**, or surprise. This notion can be formalized using the error in some prediction (of a signal or multiple signals) to compute the instantaneous *Absolute Error* or *Squared Error*. We can obtain a less noisy measure of violated expectations with a windowed average of the error, which we call *Expected Error*. Regardless of the specific form, if the error increases, then the curiosity reward increases, encouraging further sampling for that target. Such errors can be normalized, such as was done for *Unexpected Demon Error* (White et al., 2014), to mitigate the impact of the differing magnitudes of and noise in the targets.

Another category of methods focus on **learning progress**, and assume that the learning system is capable of continually improving its policy or predictions. For approaches designed for tabular stationary problems, this is trivially true: (Chentanez et al., 2005; Still and Precup, 2012; Little and Sommer, 2013; Meuleau and Bourgine, 1999; Barto and Şimşek, 2005; Szita and Lőrincz, 2008; Lopes et al., 2012). The most well-known approaches for integrating intrinsic motivation make use of rewards based on improvements in (model) error: including *Error Reduction* (Schmidhuber, 1991, 2008), and Oudeyer's model *Error Derivative Change* approach (Oudeyer et al., 2007). Improvement in the value function can as be used to construct rewards, and can be computed from the *Positive Error Part* (Schembri et al., 2007), or by tracking improvement in the value function over all states (Barto and Şimşek, 2005).

An alternative to learning progress is to reward **amount of learning**. Such rewards do not penalize errors becoming worse, and instead only measure that estimates are changing: the prediction learner is still adjusting its estimates and so is still learning. *Bayesian Surprise* (Itti and Baldi, 2006) formalizes the idea of amount of learning. For a Bayesian learner, which maintains a distribution over the weights, Bayesian Surprise corresponds to the KL-divergence between this distribution over parameters before and after the update. Other measures based on information gain have been explored (Still and Precup, 2012; Little and Sommer, 2013; Achiam and Sastry, 2017; de Abril and Kanai, 2018; Still and Precup, 2012), though they have been found to perform similarly to Bayesian Surprise (Little and Sommer, 2013). Note that several learning progress measures, discussed in the previous paragraph, can be modified to reflect amount of learning by taking the absolute value, and so removing the focus on increase rather than change. We can additionally consider non-Bayesian strategies for measuring amount of learning, including those based on how much the variance in the prediction changes—*Variance of Prediction*.

3 Experimental Design

To simulate a curious learning problem that an agent may face we introduce the drifter-distractor environment. The environment is modelled in the bandit setting and has four arms, two which respond with random noise, one which slowly drifts, and one which returns a constant. This environment simulates a common situation for learning agent: having distracting signals that will produce large prediction errors, but are ultimately unlearnable.

The ideal behaviour of a curious learning agent in this environment is to first test out all of the arms, getting a sense of what each of the signals is. After it has learned a bit about each signal it should begin to hone in on the signals that it is poor at predicting. This will initially lead it to pulling the noisy arms a fair amount. The agent should then fairly rapidly move away from pulling the noisy arms, instead focusing on the drifter arm as it has a signal that is learnable, but needs to continually be tracked to keep its predictions accurate. To measure the effectiveness of each agent we can look at the prediction error of each of its learners.

Capable learners are able to adapt to the the learnability of a signal. Previous work has not investigated the impact of capable learners in a multi-prediction setting. Here step-size adaptation methods achieve this learnability adaptation by adjusting the step size up or down. We can see the benefits on the drifter-distractor problem. Capable learners allow us to use the step size to make the noisy signals less interesting to our agent. We use Autostep (Mahmood et al., 2012) to update the agent's step size online where the reward bonus of Weight Change is given by:

$$\|w_t - w_{t+1}\|_1 = \alpha_{t,i}\|\hat{y}_{t,i} - \hat{y}_{t-1,i}\|_1 = \alpha_{t,i}|\delta_{t,i}| \tag{1}$$

Where $w_t \in \mathbb{R}$ is the weight at time t, $\alpha_{t,i} \in \mathbb{R}$ is the step size of arm i at time t, $\hat{y}_{t,i} \in \mathbb{R}$ is the prediction of arm i at time t, and $\delta_{t,i}$ is the prediction error for the agent's prediction of arm i at time t. We can see clearly that a step-size adaptation method allows us to temper the error signal by driving down the step size in unlearnable situations. This property permits us to give a curiosity reward that allows the agent to make learning progress based on the error, because it only affects the reward if the prediction is learnable.

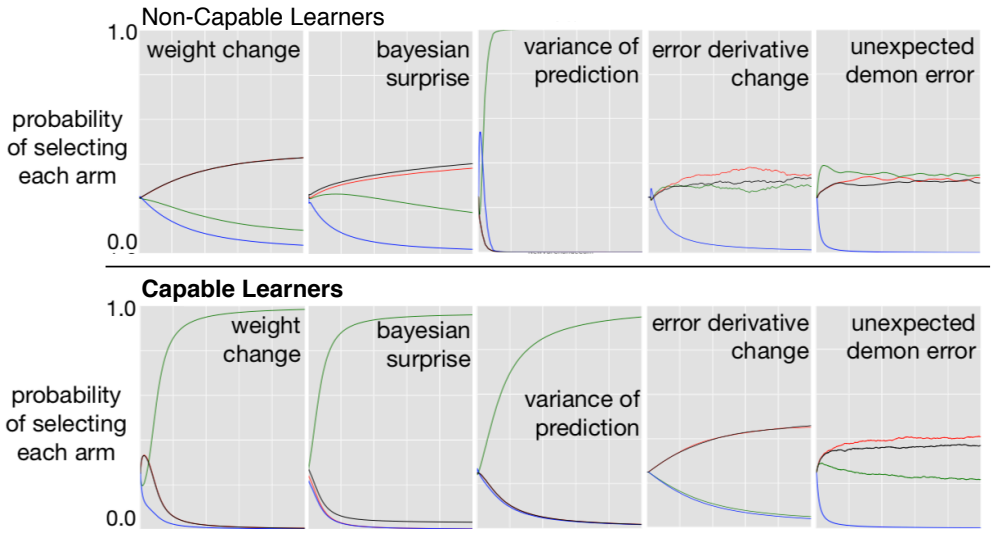


Figure 1: **The Drifter-Distractor Problem on both non-capable and capable learners.** Each subplot summarizes the learning of the control agent, over 50,000 time steps, using different curiosity rewards. Each plot shows the agent's preference for each arm over time. The drifting arm is green. We can see that the inclusion of Autostep (our capable learners) allows the simple reward function based on Weight Change and Bayesian Surprise to efficiently solve the problem. Variance of Prediction was able to solve the problem even with a weak learner, however this was due to extensive parameter sweeps and long running agents—shorter runs caused an incorrect preference for the noisy arms. The rewards based on variance result in faster preference for the drifting arm when combined with Autostep, but prefer the drifting arm less in the long run because the prediction varies less when using step-size adaptation.

4 Results

We conducted two experiments in the drifter-distractor environment, one with non-capable learners and one with capable learners. In Figure 1 we show the results for five of the methods tested in this domain. For each experiment, an extensive parameter search was conducted over the the parameters of control agent (Gradient Bandit), the prediction learners, and the reward functions.

We can see from Figure 1 that the inclusion of a capable learner allowed both the Weight Change agent and the Bayesian Surprise agent to effectively track the learnable arms while ignoring the noisy arm. Without a capable learner these agents were dominated by the error of the noisy arms and were not effective. Variance of Prediction was able to solve the problem without a capable learner—this, however, was an artifact of the parameter sweep, which chose very long running averages, enabling the larger variance to be detected. Shorter averaging windows for the variance computations caused an incorrect preference for the noisy arm.

There are two key conclusions from this experiment. First, capable learners were critical for curiosity rewards based on amount of learning, particularly Weight Change and Bayesian Surprise. Without Autostep, both Weight Change and Bayesian Surprise incorrectly cause the agent to prefer the two high-variance arms because their targets continually generate changes to the prediction. With Autostep, however, the weights converge for the constant and high-variance arms, and both agents correctly prefer the drifting arm. Second, measures based on violated expectations—Unexpected Demon Error and Error Derivative Change—either induce uniform selection or focus on noisy arms, with or without Autostep. Full results of the tested methods and domains can be found at `http://rldm.investigatingcuriosity.com`.

5 Conclusions

The goal of this work was to investigate curiosity rewards in the multi-prediction setting. This paper has three main contributions: (1) Introduce a new benchmark for curiosity-driven exploration. (2) Survey existing ideas for curiosity driven exploration and test them on our proposed benchmark. (3) Show how capable learners can allow curiosity mechanisms, specifically the step-size adapted Weight Change, to perform strongly on the proposed task.

The problem introduced in this work formalizes the multi-prediction setting as a non-stationary multi-armed bandit. This formalism allowed us to draw clear conclusions about the efficacy of existing curiosity rewards for the task of choosing what to learn about if we hope to minimize error for multiple predictions. The focus of future work is on scaling up to a larger setting with a actor-critic or SARSA behavior policy (Sutton and Barto, 2018), with a Horde (Sutton et al., 2011) of prediction and control learners rather than the LMS learners used here.

We demonstrated how this formalism can represent a variety of types of targets we expect to see in a multi-prediction setting, including those that are noisy, drifting or easy-to-predict, and investigated performance of these curiosity re-

wards across several such settings, with both weak learners and capable learners. We surveyed and studied 15 different curiosity rewards, suitable for our pure exploration setting. We reach a surprisingly clear conclusion, particularly considering the number of approaches surveyed. Simple curiosity rewards based on learner parameters, such as change in weights, can be highly effective when paired with a capable learner.

References

Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv:1703.01732*, 2017.

Andrew G Barto and Ozgür Şimşek. Intrinsic motivation for reinforcement learning systems. In *Yale Workshop on Adaptive and Learning Systems*, 2005.

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Adv. in Neural Inform. Process. Sys.*, pages 1281–1288, 2005.

I Magrans de Abril and Ryota Kanai. Curiosity-driven reinforcement learning with homeostatic regulation. *arXiv:1801.07440*, 2018.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. *arXiv:1704.03003*, 2017.

Laurent Itti and Pierre F Baldi. Bayesian surprise attracts human attention. In *Adv. in Neural Inform. Process. Systems*, 2006.

Andrew Jacobsen, Matthew Schlegel, Cameron Linke, Thomas Degris, Adam White, and Martha White. Meta-descent for online, continual prediction. 2019.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv:1611.05397*, 2016.

Alex Kearney, Vivek Veeriah, Jaden B Travnik, Richard S Sutton, and Patrick M Pilarski. TIDBD: Adapting temporal-difference step-sizes through stochastic meta-descent. *arXiv:1804.03334*, 2018.

Daniel Ying-Jeh Little and Friedrich Tobias Sommer. Learning and exploration in action-perception loops. *Front. in Neural Circuits*, 2013.

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Adv. in Neural Inform. Process. Sys.*, pages 206–214, 2012.

A. Rupam Mahmood, Richard S Sutton, Thomas Degris, and Patrick M Pilarski. Tuning-free step-size adaptation. In *ICASSP*, 2012.

Nicolas Meuleau and Paul Bourgine. Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35(2):117–154, 1999.

Joseph Modayil, Adam White, and Richard S Sutton. Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior*, 22 (2):146–160, 2014.

Pierre-Yves Oudeyer, Frédéric Kaplan, and Verena Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 2007.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv:1802.10567*, 2018.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

Massimiliano Schembri, Marco Mirolli, and Gianluca Baldassarre. Evolving childhood's length and learning parameters in an intrinsically motivated reinforcement learning robot. In *International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, 2007.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, 1991.

Jürgen Schmidhuber. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In *Workshop on Anticipatory Behavior in Adaptive Learning Systems*, 2008.

Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction, 2nd Edition*. MIT press, 2018.

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *International Conference on Autonomous Agents and Multiagent Systems*, 2011.

István Szita and András Lőrincz. The many faces of optimism: a unifying approach. In *International Conference on Machine Learning*, pages 1048–1055, 2008.

Adam White, Joseph Modayil, and Richard S Sutton. Surprise and curiosity for big data robotics. In *AAAI Workshop on Sequential Decision-Making with Big Data*, 2014.

Learned human-agent decision-making, communication and joint action in a virtual reality environment

Patrick M. Pilarski[*]
DeepMind & University of Alberta
Edmonton, Alberta, Canada

Andrew Butcher
DeepMind
Edmonton, Alberta, Canada

Michael Johanson
DeepMind
Edmonton, Alberta, Canada

Matthew M. Botvinick
DeepMind
London, UK

Andrew Bolt
DeepMind
London, UK

Adam S. R. Parker
University of Alberta
Edmonton, Alberta, Canada

Abstract

Humans make decisions and act alongside other humans to pursue both short-term and long-term goals. As a result of ongoing progress in areas such as computing science and automation, humans now also interact with non-human agents of varying complexity as part of their day-to-day activities; substantial work is being done to integrate increasingly intelligent machine agents into human work and play. With increases in the cognitive, sensory, and motor capacity of these agents, intelligent machinery for human assistance can now reasonably be considered to engage in *joint action* with humans—i.e., two or more agents adapting their behaviour and their understanding of each other so as to progress in shared objectives or goals. The mechanisms, conditions, and opportunities for skillful joint action in human-machine partnerships is of great interest to multiple communities. Despite this, human-machine joint action is as yet under-explored, especially in cases where a human and an intelligent machine interact in a persistent way during the course of real-time, daily-life experience (as opposed to specialized, episodic, or time-limited settings such as game play, teaching, or task-focused personal computing applications). In this work, we contribute a virtual reality environment wherein a human and an agent can adapt their predictions, their actions, and their communication so as to pursue a simple foraging task. In a case study with a single participant, we provide an example of human-agent coordination and decision-making involving prediction learning on the part of the human and the machine agent, and control learning on the part of the machine agent wherein audio communication signals are used to cue its human partner in service of acquiring shared reward. These comparisons suggest the utility of studying human-machine coordination in a virtual reality environment, and identify further research that will expand our understanding of persistent human-machine joint action.

Keywords: human-agent joint action, prediction learning, policy learning, emergent communication, augmented intelligence

Acknowledgements

We are deeply indebted to our DeepMind colleagues Drew Purves, Simon Carter, Alex Cullum, Kevin McKee, Neil Rabinowitz, Michael Bowling, Richard Sutton, Joseph Modayil, Max Cant, Bojan Vujatovic, Shibl Mourad, Leslie Acker, and Alden Christianson for their support, suggestions, and insight during the pursuit of the work in this manuscript.

[*]Corresponding author. This work was conducted at DeepMind, with collaboration from the University of Alberta.

> "First Thoughts are the everyday thoughts. Everyone has those. Second Thoughts are the thoughts you think about the way you think. People who enjoy thinking have those. Third Thoughts are thoughts that watch the world and think all by themselves."
>
> *Terry Pratchett, A Hat Full of Sky*

1 Understanding and Improving Human-Machine Joint Action

Humans regularly make decisions with and alongside other humans. In what has come to be defined as *joint action*, humans coordinate with other humans to achieve shared goals, sculpting both their actions and their expectations about their partners during ongoing interaction (Sebanz et al. 2006; Knoblich et al. 2011; Pesquita et al. 2018). Humans now also regularly make decisions in partnership with computing machines in order to supplement their abilities to act, perceive, and decide (Pilarski et al. 2017). It is natural to expect that joint action with machine agents might be able to improve both work and play. In situations where someone is limited in their ability to perceive, remember, attend to, respond to, or process stimulus, a machine counterpart's specialized and complementary abilities to monitor, interpret, store, retrieve, synthesize, and relate information can potentially offset or even invert these limitations. Persistent computational processes that extend yet remain part of human cognition, perhaps best described in the words of the author Terry Pratchett as "third thoughts," are evident in common tools like navigation software and calendar reminders.

Specifically with a focus on joint action, as opposed to more general forms of human-machine interaction, Moon et al. (2013), Bicho et al. (2010), Pilarski et al. (2013), Pezzulo et al. (2011), and others have provided compelling examples of fruitful human-machine coordination wherein a human and a machine work together and co-adapt in real-time joint action environments. One striking characteristic of many of these examples, and what separates them from other examples of human-machine interaction, is that they occur within *peripersonal space* (Knoblich et al. 2011)—i.e., interaction is perceived by the human to unfold continuously in the region of physical space surrounding their body and upon which they can act. While the perception of spatial and temporal proximity between partners has been shown to significantly influence joint decision making (as reviewed by Knoblich et al. 2011), peripersonal joint action settings have to date received less attention than other settings for human-machine interaction. The study of how different machine learning approaches impact human-machine joint action is even less developed, but in our opinion equally important. Our present work therefore aims to extend the discussion on how a human decision maker (here termed a *pilot*) and a machine learning assistant (termed a *copilot*) can learn together and make decisions together in service of a shared goal. We do so by defining a virtual reality environment to probe real-time joint action between humans and learning machines, and, using this environment, contribute a human-machine interaction case study wherein we demonstrate the kinds of changes that might be expected as a pilot interacts with different machine learning copilots in a shared environment.

2 Virtual Reality Environment and Protocol

A single participant engaged in a foraging task over multiple experimental blocks. This foraging task was designed so as to embed a hard-to-learn sensorimotor skill within a superficially simple protocol. In each block, the participant was asked to interact with a simulated world and a machine assistant via a virtual reality (VR) headset and two hand-

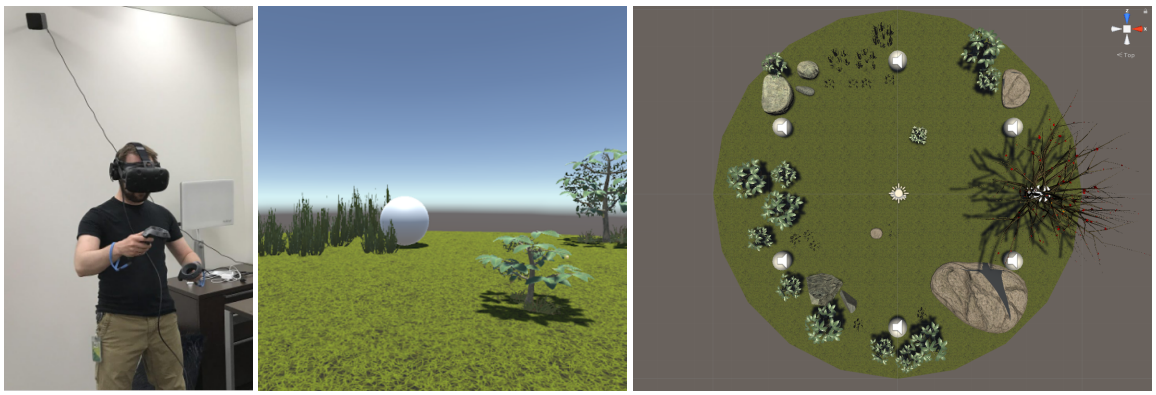


Figure 1: The virtual-reality foraging environment used to explore human-agent learning and joint action, comprised of six equidistant fruit objects, background detail, and a repeated cycle of day and night illumination. The human participant (the *pilot*) was able to move about within the virtual world and used their hand-held controllers to both harvest fruit in varying states of ripeness and train their machine-learning assistant (the *copilot*).

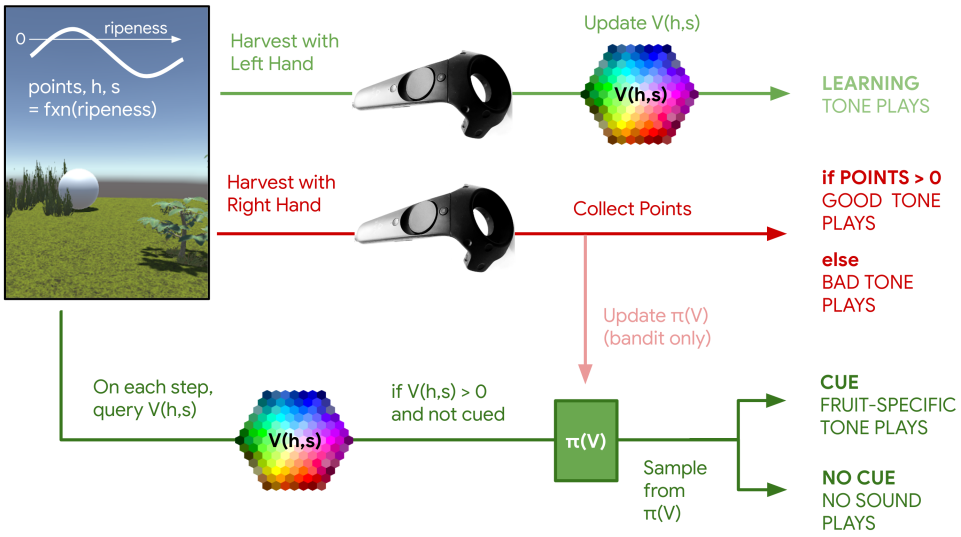


Figure 2: Schematic showing the interactions available to both the human pilot and the machine copilot. Using the right and left hand controllers, respectively, the pilot was able to harvest fruit or train their copilot to estimate the value of hue/saturation combinations, each action resulting in different sound cues as shown. On every time step, the copilot provided sound cues according to its learned predictions $V(h, s)$ and, for the bandit condition, its learned policy $\pi(V)$.

held controllers (HTC Vive with deluxe audio strap). The pilot was instructed that in each block they were to move through the world to collect objects, and that these objects would grant them "points"; they were told that, during the experiment, their total points would be reflected in visual changes to the environment, and that they would receive a unique, momentary audio cue whenever they gained or lost points as a result of their actions (and that they could also be given different audio cues in situations where they might expect to gain or lose points).

Loosely inspired by the bee foraging example of Schultz et al. (1997), the virtual world presented to the pilot was a simple platform floating in space with six coloured balls ("fruit") placed at equidistant locations around its perimeter (Fig. 1). To tease out behavioural changes in different conditions for pilot/copilot joint action, over the course of a single trial the light level in the world varied from full sunlight ("day") to regular periods of darkness lasting roughly 20s ("night", c.f., Fig. 3c, bottom trace). While copilot perception was unaffected by light level, during twilight and night phases the pilot was by design unable to determine the colour of any objects in the environment, seeing only their general shape. The main mechanic of the world was that the pilot could collect fruit to either increase their points or, in some blocks, collect fruit in order to teach their copilot.

To create conditions for skill learning on the part of the pilot, the environment was designed with a confounding factor (ripeness) that related the appearance of the fruit as observed by the pilot to the reward structure of the task. Over the course of time, each fruit underwent a "ripening cycle" wherein it cycled through a progression of colours—hue and saturation levels—and reward that varied in relation to the time since the fruit's appearance. If a fruit was not contacted before a fixed time interval by the pilot, i.e., the end of the ripening cycle, it would disappear and no point gains or losses will be credited to the participant. A short, variable time after the participant collected a fruit or the fruit disappeared due to time, a new fruit would drop from above to the same position as the previous fruit. These new fruit were assigned a random fraction between 0% and 95% through their ripening cycle.

The pilot started each trial in the centre of the platform, was able to move short distances and rotate in place, and could make contact with the fruit using either their left or right hand controllers (Fig. 2). Upon making contact with one of the fruit via their right-hand controller ("harvesting"), the ball would disappear and the pilot would hear an audio cue indicating that they either gained or lost points (one distinct sound for each case, with a small mote of light appearing in place of the fruit when points were gained). Upon contacting a ball with their left-hand controller ("teaching"), the fruit would disappear with no points gained or lost by the pilot, and they would hear a unique tone indicating that they had given information to their copilot (for blocks involving a copilot, otherwise the fruit disappeared and no sound played).

Harvesting fruit was the only way the pilot could gain points. The points gained or lost by the participant for collecting a fruit was determined by a sinusoidal reward function that varied according to the time since the beginning of the ripening cycle (Fig. 2). All fruit in a given trial were assigned the same, randomly selected reward sinusoid (in terms of phase) and related hue/saturation progression—i.e., all balls in a given trial would ripen in the same way, and would generate points in the same way, but these ripening mechanics would vary from trial to trial. The frequency of the reward sinusoid (i.e., number of reward maxima and minima until a fruit's disappearance), positive/negative reward offset, length of the ripening cycle, and time range until reappearance (all akin to difficulty), were predetermined and held constant across

all trials and blocks. Difficulty in terms of these parameters was empirically preselected and calibrated so as to provide the pilot with a challenging pattern-learning problem that was still solvable within a single trial.

Following an extended period of familiarization with the navigation and control mechanics of the VR environment and the different interaction conditions, the participant (one of the co-authors for this pilot study) experienced three experimental blocks each consisting of three 180s trials; each trial utilized previously unseen ripening mechanics in terms of colour presentation and points phase. There was an approximately 20s break between trials. The blocks related to the three different conditions (Fig. 2), presented in order, as follows:

Condition 1, No copilot (NoCP): The pilot harvested fruit without any signalling or support from a machine copilot. Using the left, training hand to contact fruit had no effect, and did not provide any additional audio cues.

Condition 2, Copilot communication via Pavlovian control (Pav): The pilot harvested fruit with support from a machine copilot that *provided audio cues in a fixed way* based on its learned predictions. As described above, the pilot was able to train the copilot by contacting fruit with their left hand controller. Practically, this amounted to updating the value function of the copilot, denoted $V(h, s)$, according to the points value associated with a fruit's current hue h and saturation level s at the time of contact; updates were done via simple supervised learning. At each point in time, the copilot queried its learned value function $V(h, s)$ with the current colour values of each of the six fruits, and, if the value of $V(h, s)$ was positive for a fruit, triggered an audio cue that was unique to that fruit—each fruit had a characteristic sound. In other words, feedback from the copilot to the human pilot was based on a pre-determined function that mapped learned predictions to specific actions (an example of Pavlovian control and communication, c.f., Modayil et al. (2014) and Parker et al. (2014)).

Condition 3, Copilot communication learned through trial and error (Bandit): The pilot harvested fruit with support from a machine copilot that *provided audio cues in an adaptable way* based on collected points (reward) and its learned predictions. This condition was similar to Condition 2 in terms of how the pilot was able to train the copilot. However, instead of deterministically playing an audio cue for the pilot each time the copilot's prediction $V(h, s)$ for a given fruit was positive, the copilot was instead presented with a decision whether or not to play an audio cue for the pilot. The decision to play a cue was based on a stochastic policy $\pi(V)$ that was updated as in a contextual bandit approach (Sutton and Barto, 2018) according to the points collected by the pilot if and when a fruit was harvested. In essence, if the copilot cued the user and this resulted in the pilot gaining points a short time later, it would reinforce the copilot's probability of playing a sound when it predicted a similar level of points in the future; should the pilot instead harvest the fruit and receive negative points, as when a fruit is harvested after the peak in its ripening cycle and/or the pilot is consistently slow to react to the copilot's cue, the copilot would decrease its probability of playing a sound when it predicted that level of expected points. The copilot's control policy used learned predictions as state, c.f., prediction in human and robot motor control (Wolpert et al. 2001; Pilarski et al. 2013).

3 Results and Discussion

Figure 3 presents the aggregate behaviour of the pilot in terms of total score over all three trials per condition, a fruit-by-fruit breakdown of total score, and a detailed presentation of time-series data from the second trial of the experiment. As a key finding, we observed that interaction with different copilots (Pav and Bandit) led to different foraging behaviours on the part of the pilot during day and night, especially as compared to the no copilot (NoCP) condition. Interactions with both the Pav and the Bandit copilot led to more foraging behaviour during night-time periods, as compared to the NoCP condition (Fig. 3c). Learning to interpret the communication from these copilots appeared to induce multiple foraging mistakes on the part of the pilot, especially during night-time phases (Fig. 3a,c) and less familiar fruit locations (Fig. 3b). Despite this, the total points collected in the absence of any mistakes (the score without any negative point value events, Fig. 3a+) suggest that collaboration with a policy-learning copilot could potentially lead to effective joint action once a good policy has been learned by both the pilot and the copilot. Teaching interactions ($\Delta V(h, s)$, Fig. 3c) also provided a useful window into pilot skill learning. Broadly, the behaviour patterns observed in this preliminary study suggest that gradual addition of cues from a copilot, as in the Bandit policy-learning condition, is likely more appropriate than a strictly Pavlovian control approach. These initial results also indicate that there is room for more complex copilot architectures that can capture the appropriate timing of cues with respect to pilot activity (e.g., a pilot harvesting wrong fruit, or hesitation as per Moon et al. (2013)), and motivate more detailed study into the impact that pilot head position, gaze direction, light level, and other relevant signals have on a copilot's ability to generate good cues. Time delays and credit assignment matter in this joint-action setting and require further thought.

Conclusions: This work demonstrated a complete (though straightforward) cycle of human-agent co-training and learned communication in a VR environment, where closing the loop between human learning (human learns then trains an agent regarding patterns in the world) and agent learning (agent learns to make predictions and provide cues that must be learned by the pilot) appears possible to realize even during brief interactions. The VR fruit foraging protocol presented in this work proved to be an interesting environment to study pilot-copilot interactions in detail, and allowed us to probe the way human-agent behaviour changed as we introduced copilots with different algorithmic capabilities.

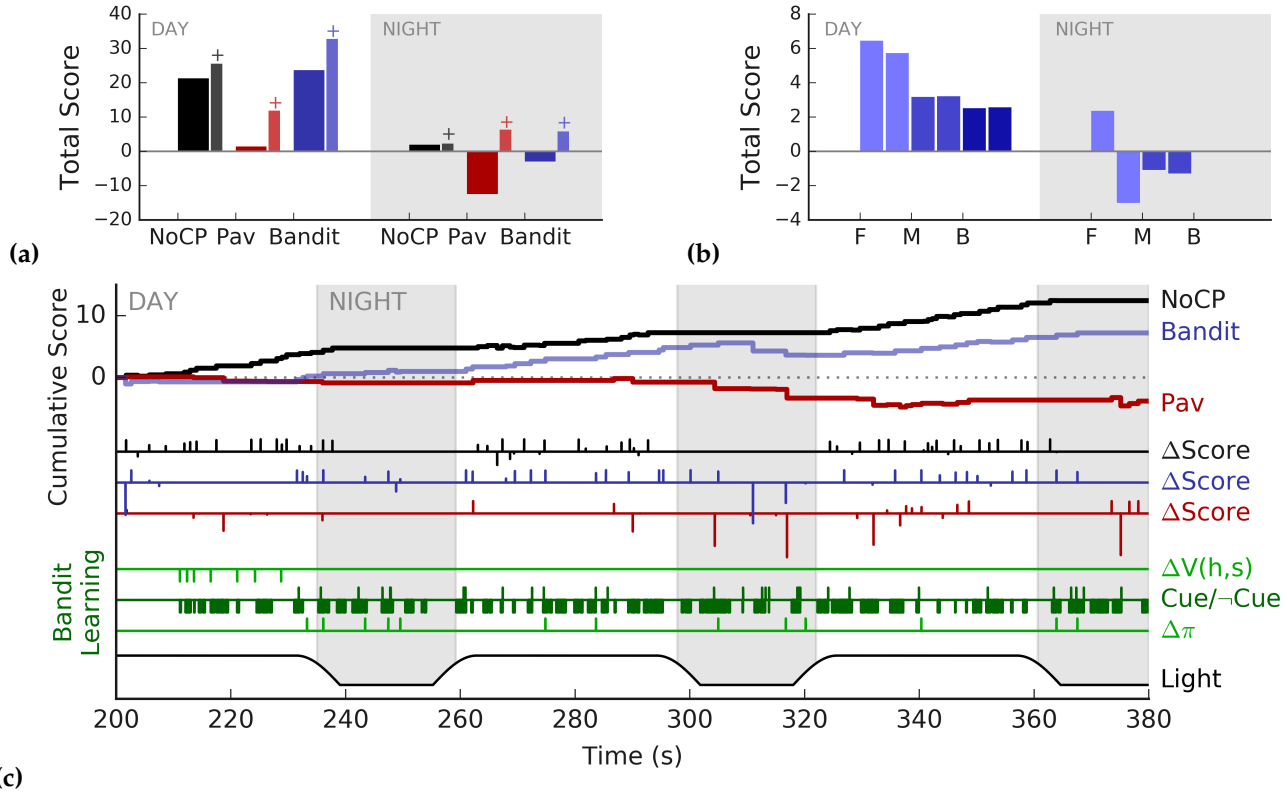


Figure 3: Results from a single pilot participant for the three experimental conditions (*NoCP, Pav,* and *Bandit*). (a) Total score acquired by the pilot in each condition during day and night, as summed over all three trials in a block, along with the total score excluding any events with negative points (+); (b) total Bandit score with respect to fruit location to the front (F), middle (M) or back (B) with respect to the pilot's starting orientation; and (c) representative example of time-series data from the second trial for all three conditions, cross-plotting cumulative score, changes in score ($\Delta Score$), light level, and bandit learning in terms of human teaching actions ($\Delta V(h,s)$), copilot decisions to cue or not cue (*up/down ticks*), and post-cue updates to the copilot's policy as a result of pilot activity ($\Delta\pi$).

References

Bicho, E., et al. (2011). Neuro-cognitive mechanisms of decision making in joint action: A human-robot interaction study. *Human Movement Science* 30, 846–868.

Knoblich, G., et al. (2011). Psychological research on joint action: Theory and data. In WDK2003 (Ed.), *The Psychology of Learning and Motivation* (Vol. 54, pp. 59-101). Burlington: Academic Press.

Modayil, J., Sutton, R. S. (2014). Prediction driven behavior: Learning predictions that drive fixed responses. *AAAI Wkshp. AI Rob.*

Moon, A., et al. (2013). Design and impact of hesitation gestures during human-robot resource conflicts. *Journal of Human-Robot Interaction* 2(3), 18–40.

Parker, A. S. R., et al. (2014). Using learned predictions as feedback to improve control and communication with an artificial limb: Preliminary findings. *arXiv*:1408.1913 [cs.AI]

Pesquita, A., et al. (2018). Predictive joint-action model: A hierarchical predictive approach to human cooperation. *Psychon. Bull. Rev.* 25, 1751–1769.

Pezzulo, G., Dindo, H. (2011). What should I do next? Using shared representations to solve interaction problems. *Exp. Brain. Res.* 211, 613–630.

Pilarski, P. M., et al. (2013). Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints. *Proc. IEEE Int. Conf. Rehab. Robotics (ICORR)*, Seattle, USA, 1–8.

Pilarski, P. M., et al. (2017). Communicative capital for prosthetic agents. *arXiv*:1711.03676 [cs.AI]

Schultz, W., et al. (1997). A neural substrate of prediction and reward. *Science* 275(5306), 1593–9.

Sebanz, N., et al. (2006). Joint action: Bodies and minds moving together. *Trends. Cogn. Sci.* 10(2), 70–76.

Sutton, R. S., Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Second Edition. Cambridge: MIT Press.

Wolpert, D. M., et al. (2001). Perspectives and problems in motor learning. *Trends Cogn. Sci.* 5(11), 487–494.

Learning Temporal Abstractions from Demonstration: A Probabilistic Approach to Offline Option Discovery

Francisco M. Garcia, Chris Nota, and Philip S. Thomas
College of Information and Computer Sciences
University of Massachusetts Amherst
{fmgarcia,cnota,pthomas}@cs.umass.edu

Abstract

The use of temporally extended actions often improves a reinforcement learning agent's ability to learn solutions to complex tasks. The options framework is a popular method for defining closed-loop temporally extended actions, but the question of how to obtain options appropriate for a specific problem remains a subject of debate. In this paper, we consider *good* options to be those that allow an agent to represent optimal behavior with minimal decision-making by the policy over options, and propose learning options from historical data. Assuming access to demonstrations of (near)-optimal behavior, we formulate an optimization problem whose solution leads to the identification of options that allow an agent to reproduce optimal behavior with a small number of decisions. We provide experiments showing that the learned options lead to significant performance improvement and we show visually that the identified options are able to reproduce the demonstrated behavior.

Keywords: Temporal abstraction; Options; Hierarchical RL

1 Introduction

Traditionally, reinforcement learning techniques focus on learning how to solve a task by only taking into consideration primitive actions that last for a single time-step. This approach makes learning complex tasks a difficult, mainly because an agent would have to make a large number of decisions to reach a state that is far from its current state. To address this difficulty, researchers have turned their attention to temporally extended actions that last for several time-steps. Their potential to deal with complex problems has led to the development of techniques for discovering or learning temporal abstractions autonomously in recent years [1, 4, 6, 9]; however, there is no consensus as to what constitutes good abstractions or how to best identify them. Nonetheless, two frameworks in particular stand out in the literature: macros [5] and options [7, 10].

One popular approach for learning abstractions is to identify specific states deemed important (subgoals) and learn policies to reach those states, [4, 6]. For example, McGovern et al. [6] proposed finding *bottleneck* states—states often found in trajectories reaching a goal state—and learn options to reach those bottlenecks discovered. Another technique is to directly learn the parameters of stochastic options based on the reward function, as the agent learns how to solve a new task [1, 3, 8]. In contrast to the methods previously described, we build on the intuition presented by Garcia et al. [2], where the authors draw a connection between data compression and generally useful temporal abstractions. They consider trajectories from optimal policies to be analogous to messages to be encoded, and the symbols available for encoding represent primitives and macros. Therefore, using compression to reduce the number of symbols that represent a trajectory leads to finding macros that reduce the number of decisions an agent must make to generate such trajectory. For example, a trajectory obtained by the sequence of actions $\{a_1, a_1, a_2, a_1\}$, where a_1 and a_2 are two primitive actions, is represented by 4 symbols. Given a macro, $m_1 = \{a_1, a_1\}$, the same sequence of actions can be represented by 3 symbols as $\{m_1, a_2, a_1\}$. The authors argue that desirable macros are those that frequently occur in optimal trajectories and correspond to the macros that can be used to represent those trajectories with small number of symbols. Under this perspective, compression techniques lead to a natural way of obtaining useful temporal abstractions.

In this paper, we extend this idea to the general setting of stochastic options. We make the observation that reducing the number of symbols needed to represent trajectories from a policy is akin to reducing the number of the decisions an agent makes to generate them. With that perspective in mind, we propose learning the options that minimize the expected number of decisions needed to represent optimal trajectories and maximize the probability of generating them.

2 Background and Notation

A *Markov decision process* (MDP) is a tuple, $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma, d_0)$, where $\mathcal{S}$ is the set of possible states of the environment, $\mathcal{A}$ is the set of possible actions that the agent can take, $P(s, a, s')$ is the probability that the environment will transition to state $s' \in \mathcal{S}$ if the agent executes action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, $R(s, a, s')$ is the real-valued reward received after taking action a in state s and transitioning to state s', d_0 is the initial state distribution, and $\gamma \in [0, 1]$ is a discount factor for rewards received in the future. We use t to index the time-step and write S_t, A_t, and R_t to denote the state, action, and reward at time t. A *policy*, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, provides a conditional distribution over actions given each possible state: $\pi(s, a) = \Pr(A_t = a | S_t = s)$. We denote a trajectory of length t as $h_t = (s_0, a_0, r_0, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$, that is h_t is defined as a sequence of states, actions and rewards observed after following some policy for t time-steps. We also use H_t to denote a random variable representing a trajectory. *Temporally extended* actions change allow an agent to execute an action composed of several primitive actions, lasting for multiple time-steps. Below we discuss in detail two common types of temporal abstraction: *macros* and *options*.

Macros are open-loop temporally extended actions; that is, sequences of actions that are executed to termination regardless of the states encountered during execution. Formally, a macro of length k is a sequence of actions $m = (a^1, a^2, \ldots, a^k)$, where a^i represents the i^{th} action in the sequence. If carefully constructed, macros can dramatically improve the speed with which an agent learns to solve a task [5]. One way for discovering macros was recently proposed by Garcia et al. [2], where the authors show how compression techniques can be used to identify repeating action patterns. They note a relationship between the number of symbols needed to represent optimal trajectories and the reusability of the macros represented by these symbols.

The options framework [10, 11] is a generalization of macros to closed-loop policies. An option, $o = (\mathcal{I}_o, \mu_o, \beta_o)$, is a tuple in which $\mathcal{I}_o \subseteq \mathcal{S}$ is the set of states in which option o can be executed (the *initiation set*), μ_o is a policy that governs the behavior of the agent while executing o, and $\beta_o : \mathcal{S} \to [0, 1]$ is a termination function that determines the probability that o terminates in a given state. We assume that $\mathcal{I}_o = \mathcal{S}$ for all options o; that is, the options are available at every state. We also consider primitive actions to be options which always terminate and select one specific action with probability one.

We build on the idea that compressing trajectories to minimize the number of decisions an agent must make leads to discovering generally useful abstraction, as described by Garcia et al. [2], and develop an analogous method for the more general case of stochastic options. We propose an off-line technique where options are learned from historical data (assumed to be obtained from (near)-optimal policies) by directly minimizing the expected number of decisions the agent makes while simultaneously maximizing the probability of generating the observed trajectories.

3 Offline Option Learning

To generalize the idea of using compression to identify macros to the setting of stochastic options, we first observe that the use of a symbol in the compressed representation of a trajectory corresponds to the agent executing a different primitive or macro. Therefore, compressing a trajectory results in minimizing the number of decisions the agent has to make. In order to learn options from a set of trajectories, we minimize the expected number of decisions needed to generate the observed trajectories while maximizing the probability of generating them.

3.1 Problem Formulation

Notice that that while solving a task, at every time-step, t, there was an option that was used to execute action A_t in state S_t. We will consider primitive actions to be options which always terminate and select one specific action with probability one; that is, for an option o corresponding to a primitive a the termination function would be given by $\beta_o(s) = 1, \forall s \in \mathcal{S}$ and the policy by $\mu(s, a') = 1$ if $a' = a$ and $\mu(s, a') = 0$ otherwise, $\forall s \in \mathcal{S}$. Let T_t be a Bernoulli random variable, where $T_t = 1$ if the executing option terminates at the t^{th} state in a trajectory H, and $T_t = 0$ if it did not. This implies that if $T_t = 1$, the agent would have to pick a new option at state s_t. If $T_t = 0$, the agent does not pick a new option. Note that the distribution of T_t depends on the set of available options and a policy over options. Let $\mathcal{O}$ denote a set of options, $\{o_1, \ldots, o_n\}$, and let H be random variable denoting a trajectory of length $|H|$ generated by a near-optimal policy and H_t a random variable denoting the sub-trajectory of H up to the t^{th} state. We seek to find a set, $\mathcal{O}^* = \{o_1^*, \ldots, o_n^*\}$, that maximizes the following objective: $J(\pi, \mathcal{O}) = \mathbf{E}\big[\sum_{t=1}^{|H|} \Pr(T_t = 0, H_t|\pi, \mathcal{O}) + \lambda_1 g(H, \mathcal{O})\big]$, where $g(h, \mathcal{O})$ is a regularizer that encourages a diverse set of options, and λ_1 is a scalar hyperparameter. One choice is the KL divergence: $g(h, \mathcal{O}) = \frac{1}{n} \sum_{o,o' \in \mathcal{O}} \sum_{t=0}^{|h|-1} D_{\text{KL}}\left(\mu_o(s_t) || \mu_{o'}(s_t)\right)$. If we are also free to learn the parameters of π, then $\mathcal{O}^* \in \arg\max_{\mathcal{O}} \arg\max_{\pi} J(\pi, \mathcal{O})$.

Intuitively, we seek to find options that terminate as infrequently as possible while still generating near-optimal trajectories with high probability. Given a set of options and a policy over options, we can calculate these probabilities *exactly*, allowing us to optimize this objective directly using gradient-based methods. We can improve our results by averaging over a *set* of near-optimal trajectories, rather than a single trajectory. In the next section, we show how these probabilities can be calculated and present a slightly modified form of the objective that is more useful in practice.

3.2 Optimization Objective for Learning Options

We can approximate the objective J from a set of sample trajectories $\mathcal{H}$ obtained from optimal policies. Because the probability of generating any trajectory approaches 0 as the length of the trajectory increases, we make a slight modification to the original objective that leads to better numerical stability. We explain these modifications after introducing the objective $\hat{J}$, which we optimize in practice:

$$\hat{J}(\pi, \mathcal{O}) = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \underbrace{\lambda_2 \Pr(H = h|\pi, \mathcal{O})}_{\text{Probability of trajectory}} - \underbrace{\frac{\sum_{t=1}^{|h|} \mathbf{E}\left[T_t = 1 | H_t = h_t, \pi, \mathcal{O}\right]}{|h|}}_{\text{Expected number of terminations}} + \underbrace{\lambda_1 g(h, \mathcal{O})}_{\text{Regularization term}} . \tag{1}$$

Equation 1 is derived from J by expanding the joint probability into the product of $\Pr(H_t = h_t|\pi, \mathcal{O})$ and $\Pr(T_t = 0|H_t = h_t, \pi, \mathcal{O})$. With some algebraic manipulation, we observe that maximizing $\Pr(H = h|\pi, \mathcal{O})$ while minimizing $\sum_{t=1}^{|h|} \Pr(T_t = 1|H_t = h_t, \pi, \mathcal{O})$ maximizes the objective in J. Finally, we also introduce a scalar weight λ_2 to balance the contribution of each term to $\hat{J}$. Equation 1 can be expressed entirely in terms of π, options $\mathcal{O} = \{o_1, \ldots, o_n\}$ and the transition function, P, while removing the expectations and probabilities—this allows us to compute its value exactly. These expressions are given in recursive form by the following theorems:

Theorem 1. *Let O_t denote the option selected for execution at the t^{th} state in a trajectory h_t. Given a set of options $\mathcal{O}$ and a policy over options π, the expected number of terminations is given by:* $\sum_{t=1}^{|h|} \mathbf{E}\left[T_t = 1 | H_t = h_t, \pi, \mathcal{O}\right] = \sum_{t=1}^{|h|} \left(\sum_{o \in \mathcal{O}} \beta_o(s_t) \frac{\mu_o(s_{t-1}, a_{t-1}) \Pr(O_{t-1} = o | H_{t-1} = h_{t-1}, \pi, \mathcal{O})}{\sum_{o' \in \mathcal{O}} \mu_{o'}(s_{t-1}, a_{t-1}) \Pr(O_{t-1} = o' | H_{t-1} = h_{t-1}, \pi, \mathcal{O})} \right)$,

where $\Pr(O_{t-1} = o|H_{t-1} = h_{t-1}, \pi, \mathcal{O}) = \Bigg[\Big(\pi(s_{t-1}, o) \beta_o(s_{t-1}) \Big) + \Big(P(s_{t-2}, a_{t-2}, s_{t-1}) \mu_o(s_{t-2}, a_{t-2}) \Pr(O_{t-2} = o | H_{t-1} = h_{t-1}, \pi, \mathcal{O})(1 - \beta_o(s_{t-1})) \Big) \Bigg]$, *and* $\Pr(O_0 = o|H_1 = h_1, \pi, \mathcal{O}) = \pi(s_0, o)$.

Theorem 2. *Given a set of options $\mathcal{O}$ and a policy over options π, the probability of generating a trajectory h is given by:*

$\Pr(H = h|\pi, \mathcal{O}) = d_0(s_0)\left[\sum_{o\in\mathcal{O}} \pi(s_0, o)\mu_o(s_0, a_0)f(h, o, 1)\right]\prod_{k=0}^{|h|-1} P(s_k, a_k, s_{k+1})$*, where f is the recursive function*

$$f(h,o,i) = \begin{cases} 1, & \text{if } i = |h| \\ \Big[\beta_o(s_i)\sum_{o'\in\mathcal{O}} \pi(s_{i+1}, o')\mu_{o'}(s_{i+1}, a_{i+1})f(h, o', i+1) & \\ +(1-\beta_o(s_i))\mu_o(s_{i+1}, a_{i+1})f(h, o, i+1)\Big], & \text{otherwise} \end{cases}$$

Assuming a parametric representation for $(\mu_o, \beta_o), \forall o \in \mathcal{O}$, and for a policy over option π, we are able to differentiate the objective in $\hat{J}$ with respect to their parameters and optimize with any standard optimization technique.

3.3 Learning Options Incrementally

One common issue in option discovery is identifying how many options are needed for a given problem. Oftentimes this number is pre-defined by the user based on intuition. In such a scenario, one could learn options by simply optimizing the proposed objective in Eq. 1. Instead, we propose not only learning options, but also the number of options needed, by the procedure shown in Algorithm 1.

Algorithm 1 Option Learning Framework

```
Collect set of trajectories H
Initialize option set O with primitive options
done = false
J_prev = −∞
while done == false do
    Initialize new option o' = (μ', β')
    O' = O ∪ o'
    Initialize policy π over O'

    for epoch=1,...,N do
        maximize Ĵ w.r.t o' and π
    end
    if Ĵ − J_prev < Δ then
        done = true
    else
        O = O ∪ o'  J_prev = Ĵ
    end
end
```

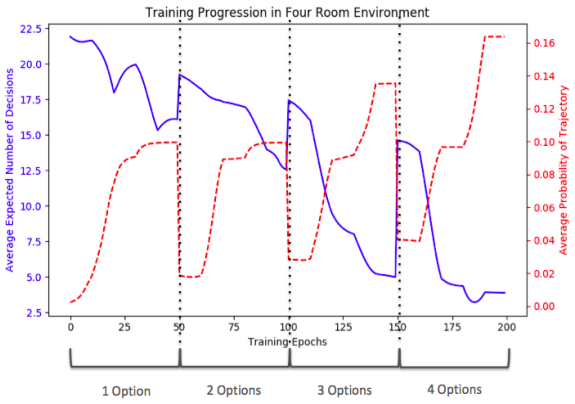
The algorithm introduces one option at a time and alternates training between π and the newly introduced option, o', for N epochs. Any previously introduced option is kept fixed. After the new option is trained, we measure how much J has improved; if it fails to improve above some threshold Δ, the procedure terminates. This results in a natural way of obtaining an appropriate number of options to represent the observed trajectories.

4 Experimental Results

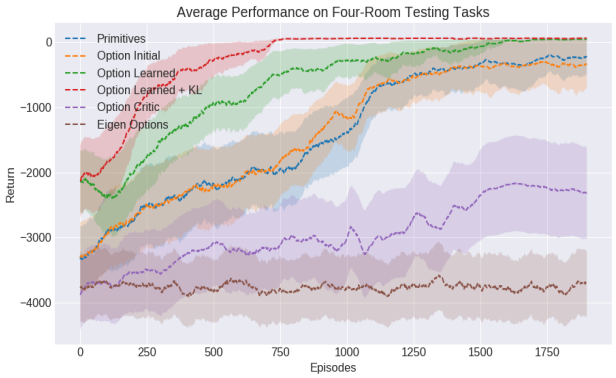
We tested our approach in the four-room domain: a Gridworld of size 40×40, in which the agent is placed in a randomly selected start state and needs to reach a randomly selected goal state. At each time-step the agent executes one of four available actions: moving left, right, up or down, and receives a reward of -1. Upon reaching the goal state, the agent receives a reward of $+10$. To generate the data needed to apply our technique, we collected ten sample trajectories from optimal policies for ten randomly placed start and goal locations. Options were represented as 2-layer neural networks with a softmax output layer over the four possible actions representing μ, and a separate sigmoid layer representing β. We used the tabular form of Q-learning with ϵ-greedy exploration as the learning algorithm, and compared our approach to two competing methods: the option-critic architecture [1] and eigenoptions [4].

Figure 1a shows the change in the loss while learning options, as new options are introduced and adapted to the sampled trajectories. It shows how the expected number of decision (blue) decreases and the probability of generating the observed trajectories (red) increases as training progresses. In this case, options were learned over the 10 sampled optimal trajectories and every 50 epochs a new option was introduced to the option set $\mathcal{O}$, for a total of 4 options. For every new option, the change in probability of generating the observed trajectories as well as the change in expected number of decisions reaches a plateau after 30 or 40 training epochs. When a new option is introduced, there is a large jump in the loss because a new policy π is initialized arbitrarily to account for the new option set being evaluated. However, after training the new candidate option, the overall loss improves beyond what it was possible before introducing it.

In Figure 1b we present the learning performance on 20 novel test MDPs (randomly selected start and goal states) after each competing method was allowed to learn options on 10 train tasks from where optimal trajectories were obtained. We contrast the performance of learned options without KL regularization, with $\lambda_1 = 0$ (green), options with KL regularization, with $\lambda_1 = 0.1$ (red), options before learning (orange), only primitives (blue), eigen options (brown), option-critic (purple). The plot shows the average return (and standard error) over 20 different start and goal locations on the y-axis and the training episode on the x-axis. Our approach was able to identify options that, given a fixed number of episodes used for training, allowed the agent to reach an optimal performance level when other methods failed to do so. This difference became even more pronounced diversity was encouraged by using a KL regularization term. These results provide compelling evidence that the learned options are efficient at solving new problems.



(a) Visualization of objective in four rooms over 200 training epochs. Every 50 epochs a new option is introduced; decreasing the expected number of decisions (blue) and increasing the probability of generating the observed trajectories (red).



(b) Performance Comparison on four rooms among option critic (purple), eigen options (brown), primitives (blue), initial options (orange), learned options without KL regularization (green), and learned options with KL regularization (red).

Figure 1: Results on four room environment. The figure on the left shows the evolution of the training loss as new options are introduced, the figure on the right shows the learning curves on test domains.

5 Conclusion and Future Work

In this work we presented an optimization objective to learn options offline from historical data. We assume that trajectories are obtained from (near)-optimal policies and learn options that allow an agent to reproduce them while minimizing the number of decisions the agent makes in order to do so. Our initial results show that options adapt to the trajectories given and they lead to more efficient learning. There are some clear direction for future development. First, more experimentation is needed to asses how well this approach scales to more complex environments. Second, our approach is only applicable for learning options offline; one interesting modification would be able to sample trajectories as the agent is learning a task and apply the procedure online to find new, better policies. This preliminary work indicates that the proposed optimization problem leads to the discovery of efficient options.

References

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2017.

[2] Francisco M. Garcia, Bruno C. da Silva, and Philip S. Thomas. Identifying reusable macros for efficient exploration via policy compression. *CoRR*, 2017.

[3] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *AAAI*, 2018.

[4] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. A Laplacian Framework for Option Discovery in Reinforcement Learning. *CoRR*, 2017.

[5] A. McGovern and R. Sutton. Macro actions in reinforcement learning: An empirical analysis. Technical report, University of Massachusetts - Amherst, Massachusetts, USA, 1998.

[6] Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, San Francisco, CA, USA, 2001.

[7] Doina Precup. *Temporal Abstraction in Reinforcement Learning*. PhD thesis, 2000.

[8] Jette Randløv. Learning macro-actions in reinforcement learning. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, NIPS'98, Cambridge, MA, USA, 1998.

[9] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*, London, UK, UK, 2002.

[10] Richard S. Sutton and Doina Precup. Intra-option learning about temporally abstract actions. In *In Proceedings of the 15th International Conference on Machine Learning (ICML-1998)*, 1998.

[11] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

Unicorn: Continual learning with a universal, off-policy agent

Daniel J. Mankowitz[†] **Augustin Žídek**[*] **André Barreto**[*] **Dan Horgan**[*] **Matteo Hessel** [*] **John Quan**[*]

Junhyuk Oh[*‡] **Hado van Hasselt**[*] **David Silver**[*] **Tom Schaul**[*]

Abstract

Some real-world domains are best characterized as a single task, but for others this perspective is limiting. Instead, some tasks continually grow in complexity, in tandem with the agent's competence. In *continual learning*, also referred to as lifelong learning, there are no explicit task boundaries. As learning agents have become more powerful, continual learning remains one of the frontiers that has resisted quick progress. To test continual learning capabilities we consider a challenging 3D domain with an implicit sequence of tasks and sparse rewards. We propose a novel Reinforcement Learning agent architecture called *Unicorn*, which demonstrates strong continual learning and outperforms several baseline agents on the proposed domain. The agent achieves this by jointly representing and learning multiple policies efficiently, using a parallel off-policy learning setup.

Keywords: Reinforcement Learning, Universal Value Function Approximators, Continual Learning

[*]DeepMind. [†]Technion Israel Institute of Technology. [‡]University of Michigan. Correspondence to: Daniel J. Mankowitz dmankowitz@google.com

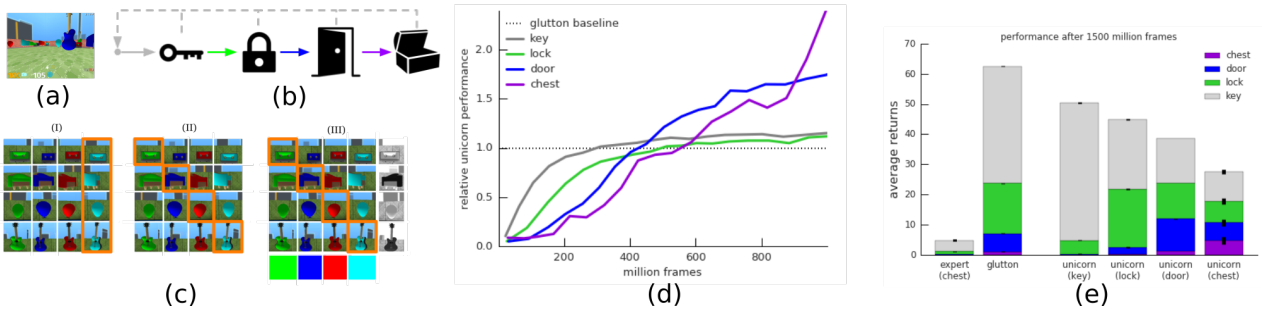


Figure 1: **(a)**: Treasure World; **(b)**: The agent needs to find and pick up objects in the specific order `key`, `lock`, `door`, `chest`. Any deviation from that order resets the sequence (dashed gray lines); **(c)**: The (I) off-policy, (II) zero-shot transfer and (III) zero-shot transfer with augmented shapes and colors setup; **(d)**: The objective is to pick up as many chests as possible (purple). The bottom left plot shows performance relative to final performance of the best baseline (glutton, dashed line) as a function of training. Our method quickly learns to become competent on all 4 subtasks, one at a time. The margin of improvement over the baseline is largest on the hardest subtasks (see section 4 for details). **(e)**: The stacked bar plot shows the average number of collected objects after training (1.5B frames). E.g., the Unicorn's `chest` policy typically collects 10 `key`s, 7 `lock`s, 6 `door`s and 5 `chest`s.

1 Introduction

Continual learning, that is, learning from experience about a continuing stream of tasks in a way that exploits previously acquired knowledge or skills, has been a longstanding challenge to the field of AI [13, 7]. An ideal continual learning agent should be able to (A) solve *multiple* tasks, (B) exhibit synergies when tasks are *related*, and (C) cope with deep *dependency* structures among tasks (e.g., a lock can only be unlocked after its key has been picked up).

Previous work on continual learning with Reinforcement Learning (RL), and specifically on solving tasks with deep dependency structures, has typically focused on *separating learning into two stages* [9, 11, 3]. In contrast, we aim to solve tasks with deep dependency structures using RL in a *single-stage end-to-end learning* setup. In addition, we aim to train the agent on *all* tasks simultaneously regardless of their complexity. We extend Universal Value Function Approximators (UVFAs) [6, 8, 10] with off-policy learning [8, 12] about multiple goals simultaneously, and scale them up to a parallel agent architecture [5, 2] and train them end-to-end. Our resulting novel continual learning agent, called *Unicorn*[1], is capable of consistently solving continual learning tasks with deep dependency structures, at scale, in complex domains (Figure 1e). Additional contributions include adapting a parallelized policy-based state-of-the-art RL architecture [2] to value-based as well as incorporating off-policy learning and off-policy corrections into the architecture. We present a continual learning experiment whereby the Unicorn learns to solve tasks with deep dependency chains (e.g., collect a key, unlock a lock, open a door, and collect a chest in that order. Then receive a reward upon completion of the task). This was impossible for existing methods (see the 'expert(chest)' baseline in Figure 1e). We also present a detailed investigation with multiple ablation experiments, showing that Unicorn effectively learns multiple tasks in parallel and exhibits synergies when tasks are related.

2 Background

Reinforcement learning (RL) A Markov decision process is defined as a 5-tuple $\langle \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma \rangle$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $r : \mathcal{S} \times \mathcal{A} \to R$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition probability distribution and $\gamma \in [0, 1)$ is a discount factor. Action value functions $Q^\pi(s, a) = E^\pi[R_t | s_t = s, a_t = a]$ estimate the expected return for an agent that selects an action $a \in A$ in some state $s \in S$, and follows policy π thereafter. We define the n-step return as $G_t^{(n)} = \sum_{k=1}^{n} \gamma^{k-1} r_{t+k} + \gamma^n \max_a Q(s_{t+n}, a)$.

Universal Value Function Approximators (UVFA) extend value functions to be conditional on a goal signal $g \in \mathcal{G}$, with their function approximator (such as a deep neural network) sharing an internal, goal-independent representation of the state $f(s)$ [6]. As a result, a UVFA $Q(s, a; g)$ can compactly represent multiple policies by conditioning on any goal signal g and choosing actions greedily. UVFA's have previously been implemented in a two-stage process involving a matrix factorization step to learn embeddings and a separate multi-variate regression procedure. In contrast, the Unicorn learns $Q(s, a; g)$ end-to-end, in a joint parallel training setup with off-policy goal learning and corrections.

Tasks vs. goals. For the purposes of this paper, we assign distinct meanings to the terms **task** (τ) and **goal signal** (g). A goal signal modulates the behavior of an agent (e.g., as input to the UVFA). In contrast, a task defines a pseudo-reward r_τ (e.g., $r_{key} = 1$ if a `key` was collected and 0 otherwise). During learning, a vector containing all pseudo-rewards is

[1] *Unicorn* stands for "UNIversal Continual Off-policy Reinforcement learNing".

visible to the agent on each transition, even if it is pursuing one specific goal. Each experiment defines a discrete set of K tasks $\{\tau_1, \tau_2, \ldots, \tau_K\}$. In transfer experiments, tasks are split between K' training tasks and $K - K'$ hold-out tasks.

3 Unicorn

This section introduces the *Unicorn* agent architecture with the following properties to facilitate continual learning. *(A):* The agent should have the ability to simultaneously learn about *multiple tasks*. We use a joint parallel training setup with a single learner but many actors working on different tasks to accomplish this (sections (II) and section (IV)). *(B):* As the agent accumulates more knowledge, we want it to generalize by reusing some of its knowledge to solve related tasks. This is accomplished using a single UVFA to capture knowledge about all tasks, with a separation of goal-dependent and goal-independent representations to *facilitate transfer* (section (I)). *(C):* The agent should be effective in domains where tasks have a *deep dependency structure*. This is the most challenging aspect, but is enabled by off-policy learning from experience across all tasks (section (III)).

(I) Value function architecture: A key component of the Unicorn agent is a UVFA, which is an approximator, such as a neural network, that learns to approximate $Q(s, a; g)$. The power of this approximator lies in its ability to be conditioned on a goal signal g. This enables the UVFA to learn about multiple tasks simultaneously where the tasks themselves may vary in their level of difficulty (e.g., tasks with deep dependencies). Our proposed UVFA architecture is depicted schematically in Figure 2 (Bottom Right): The output of the LSTM is concatenated with an "inventory stack" to form a goal-independent representation of state $f(s)$. This vector is then concatenated with a goal signal g and fed into a multi-layer perceptron (MLP) to produce the output vector of Q-values (one for each possible action $a \in \mathcal{A}$). The union of trainable parameters from all these components is denoted by θ.

(II) Behaviour policy: At the beginning of each episode, a goal signal g_i is sampled uniformly, and is held constant for the entire episode. The policy executed is ϵ-greedy after conditioning the UVFA on the current goal signal g_i: with probability ϵ the action taken a_t is chosen uniformly from $\mathcal{A}$, otherwise $a_t = \arg\max_a Q(s_t, a; g_i)$.

(III) Off-policy multi-task learning: Another key component of the Unicorn agent is its ability to learn about multiple tasks off-policy. Therefore, even though it may be acting on-policy with respect to a particular task, it can still learn about other tasks from this shared experience in parallel. Concretely, when learning from a sequence of transitions, Q-values are estimated for all goal signals g_i in the training set and n-step returns $G^{(n)}_{t,i}$ are computed for each corresponding task τ_i as $G^{(n)}_{t,i} = \sum_{k=1}^{n} \gamma^{k-1} r_{\tau_i}(s_{t+k}, a_{t+k}) + \gamma^n \max_a Q(s_{t+n}, a; g_i)$. When a trajectory is generated by a policy conditioned on one goal signal g_i (the on-policy goal with respect to this trajectory), but used to learn about the policy of another goal signal g_j (the off-policy goal with respect to this trajectory), then there are often action mismatches, so the off-policy multi-step bootstrapped targets become increasingly inaccurate. Following [14], we therefore *truncate* the n-step return by bootstrapping at all times t when the taken action does not match what a policy conditioned on g_j would have taken,[2] i.e. whenever $a_t \neq_a Q(s_t, a; g_j)$. The network is updated with gradient descent on the sum of TD errors across tasks and unrolled trajectory of length H (and possibly a mini-batch dimension B), yielding the squared loss $\mathcal{L} = \frac{1}{2} \sum_{i=1}^{K'} \sum_{t=0}^{H} \left(G^{(n)}_{t,i} - Q(s_t, a_t; g_i) \right)^2$ where errors are not propagated into the targets $G^{(n)}_{t,i}$.

(IV) Parallel agent implementation: We employ a parallel agent setup consisting of multiple *actors*, running on separate (CPU) machines, that generate sequences of interactions with the environment, and a single *learner* (GPU machine) that pulls this experience from a queue, processes it in mini-batches, and updates the value network (see Figure 2, Top Right). Each actor continuously executes the most recent policy for some goal signal g_i. Together they generate the experience that is sent to the learner in the form of trajectories of length H, which are stored in a global queue. The learner batches up trajectories pulled from the global queue (to exploit GPU parallelism), passes them through the network, computes the loss (Section (III)), updates the parameters θ, and provides the most recent parameters θ to actors upon request.

4 Experiments

Following our stated motivation for building continual learning agents, we set up a number of experiments that test the Unicorn's capability to solve (A) multiple (Section 4A), (B) related (Section 4B) and (C) dependent tasks (Section 4C).

Domain: We developed a visually rich 3D navigation domain within the DM Lab framework [1] which we call *Treasure World* (Figure 1a). The specific level used consists of one large room filled with 64 objects of multiple types and equal frequency. Whenever an object is collected, it respawns at a random location in the room. Episodes last for 60 in-game seconds, which corresponds to 450 time-steps. The continual learning experiments last for 120 in-game seconds. The objects used in the multi-task and transfer domains are different color variations of cassettes, chairs, balloons and guitars. For continual learning, the TV, ball, balloon and cake objects play the *functional* roles of a `key`, `lock`, `door` and `chest` respectively. Visual observations are provided only via a first-person perspective, and are augmented with an

[2] Returns are also truncated for the on-policy goal when epsilon (i.e., non-greedy) actions are chosen.

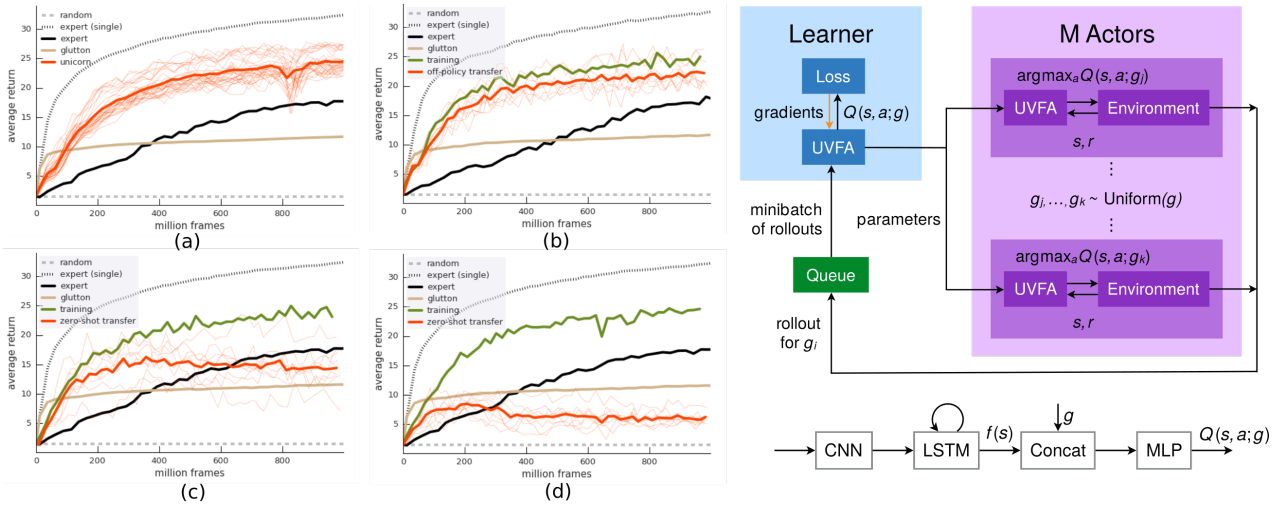


Figure 2: **(a) Multi-task Learning** A single Unicorn agent learns to collect any out of 16 object types in the environment. Each thin red line corresponds to the on-policy performance for one such task; the thick line is their average. We observe that performance across all tasks increases together, and much faster, than when learning separately about each task (black). See text for the baseline descriptions. **(b)** Off-policy ablative experiment; **(c)** Zero shot performance with augmented tasks and **(d)** Zero shot performance without augmented tasks. **Top Right:** Unicorn Actor/Learner setup **Bottom Right:** Network architecture.

inventory stack with the five most recently collected objects. Picking up is done by simply walking into the object. There is no special pick-up action; however, pick-ups can be conditional, e.g., a `lock` can only be picked up if the `key` was picked up previously. The goal signals are pre-defined one-hot vectors unless otherwise stated.

Baselines We compare four baselines to the Unicorn agent. The first two baselines have the same architecture and training setup as the Unicorn but are conditioned and learn about a single goal g. We train one of these single-task expert agents for each individual task. Baseline (1): *expert (single)*, which is the single-task expert performance averaged across all tasks. The horizontal axis for this baseline is not directly comparable as the experts together consume K times more experience than the Unicorn (as each single-task expert is trained on a separate network). We therefore represent this baseline with a dotted line to indicate an upper performance bound. Baseline (2): *expert*, which focuses on sample complexity and takes all accumulated experience of all the single-task expert agents, across all tasks, into account. In this case, the axes are directly comparable. Baseline (3): *glutton*, also uses the same architecture and training setup, but uses a single composite task whose pseudo-reward is the sum of rewards of all the other tasks $r_{glutton}(s,a) = \sum_i^K r_i(s,a)$. This is also a single-task agent that always acts on-policy according to this cumulative goal. Its performance is measured by calculating the rewards the glutton policy obtains on the individual tasks. This baseline is directly comparable in terms of compute and sample complexity. Baseline (4): a uniformly *random* policy.

(A) Learning multiple tasks: The multi-task Treasure World experiment uses 16 unique objects types (all objects have one of 4 colors and one of 4 shapes), with an associated task τ_i for each of them: picking up that one type of object is rewarding, and all others can be ignored. Figure 2a shows the learning curves for Unicorn and how they relate to the baselines; data is from two independent runs. We see that learning works on all of the tasks (small gap between the best and the worst), and that learning about all of them simultaneously is more sample-efficient than training separate experts, which indicates that there is beneficial generalization between tasks. As expected, the final Unicorn performance is much higher than that of the glutton baseline.

(B) Generalization to related tasks: The first transfer experiment (Figure 1c(II)) investigates zero-shot transfer to a set of four hold-out tasks (objects within the orange boxes) that see neither on-policy experience nor learning updates. Generalization happens only through the relation in how goal signals are represented: each $g_i \in R^8$ is a two-hot binary vector with one bit per color and one bit per shape. Successful zero-shot transfer would require the UVFA to factor the set of tasks into shape and color, and interpret the hold-out goal signals correctly. Figure 2d shows the average performance of the hold-out tasks, referred to as *zero-shot*, compared to the training tasks and the additional baselines. We observe that there is some amount of zero-shot transfer, because the zero-shot policy is clearly better than random.

The second transfer experiment (Figure 1c(III)) augments the set of training tasks by 8 abstract tasks (20 training tasks in total) where reward is given for picking up any object of one color (independently of shape), or any object of one shape (independently of color). Their goal signals are represented as one-hot vectors $g_i \in R^8$. Figure 2c shows that this

augmented training set substantially helps the zero-shot performance, above what the glutton baseline can do. These results are consistent with those of [4]. More detailed learning curves can be found in the appendix.

Ablative study - Learning only from off-policy updates: In a probing experiment (Figure 1c(I)), the Unicorn actors only act on-policy with respect to the 12 training goal signals (all non-`cyan` objects), but learning happens for the full set of 16 objects; in other words the `cyan` objects (surrounded by the orange bounding box) form a partial hold-out set, and are learned purely from off-policy experience. Figure 2b summarizes the results, which show that the agent can indeed learn about goals from purely off-policy experience. Learning is not much slower than the on-policy goals.

(C) Continual learning with deep dependencies This section presents experiments that test the Unicorn agent's ability to solve tasks with deep dependency structures (C). For this, we modified the Treasure World setup slightly: it now contains four copies of the four[3] different object types, namely `key`, `lock`, `door` and `chest`, which need to be collected in this given order (see Figure 1b). The vector of pseudo-rewards corresponding to these tasks are conditioned on precise sequences in the inventory: to trigger a reward for the `door` task, the previous two entries must be `key` and `lock`, in that order. Any object picked up out of order breaks the chain, and requires starting with the `key` again. Here, improving the competence on one task results in easier exploration or better performance on the subsequent one as shown in Figure 1d. The bar-plot in Figure 1e shows this stark contrast: compare the height of the violet bars (that is, performance on the `chest` task) for 'unicorn(`chest`)' and 'expert(`chest`)' – the Unicorn's continual learning approach scores 4.75 on average, while the dedicated expert baseline scores 0.05, not better than random. Across 5 independent runs, the `chest` expert baseline was never better than random. On the other hand, the glutton baseline learned a lot about the domain: at the end of training, it collects an average of 38.72 `key`, 16.64 `lock`, 6.05 `door` and 1.05 `chest` rewards. As it is rewarded for all 4 equally, it also encounters the same kind of natural curriculum, but with different incentives: it is unable to prioritize `chest` rewards. In contrast, the Unicorn conditioned on $g_{\texttt{chest}}$ collects an average of 9.93 `key`, 6.99 `lock`, 5.92 `door` and 4.75 `chest` rewards at the end of training. A video of the Unicorn performance in the 16 object setup is available online[4].

5 Conclusion

We have presented the Unicorn, a novel agent architecture that exhibits the core properties required for continual learning. Unicorn is able to (A) efficiently learn about multiple tasks, (B) leverage learned knowledge to solve related tasks, even with zero-shot transfer, and (C) solve tasks with deep dependencies. All of this is made efficient by off-policy learning about multiple tasks simultaneously, using parallel streams of experience coming from a distributed setup. Experiments in a rich 3D environment indicate that the Unicorn clearly outperforms the corresponding single-task baseline agents, scales well, and manages to exploit the natural curriculum present in the set of tasks.

References

[1] Beattie et. al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[2] Espeholt et. al. Scalable distributed deep-rl with importance weighted actor-learner architectures. *ICML*, 2018.

[3] Finn et. al. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML*, 2017.

[4] Hermann et. al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

[5] Mnih et. al. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[6] Schaul et. al. Universal value function approximators. In *ICML*, 2015.

[7] Schaul et. al. The barbados 2018 list of open issues in continual learning. *NIPS workshop on Continual Learning*, 2018.

[8] Sutton et. al. Horde. In *AAMAS*, 2011.

[9] Tessler et. al. A deep hierarchical approach to lifelong learning in minecraft. *AAAI*, 2017.

[10] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[11] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. *arXiv preprint arXiv:1706.05064*, 2017.

[12] Jing Peng and Ronald J Williams. Incremental multi-step q-learning. In *Machine Learning Proceedings 1994*, pages 226–232. Elsevier, 1994.

[13] Mark Bishop Ring. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin, 1994.

[14] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, 1989.

[3] With only 16 (respawning) objects in total, this is less dense than in the experiments above, in order to make it less likely that the agent collects an out-of-sequence object by mistake.

[4] https://youtu.be/h4lawNq2B9M

Penalty-Modified Markov Decision Processes: Efficient Incorporation of Norms into Sequential Decision Making Problems

Stephanie Milani *
Department of CSEE
University of Maryland, Baltimore County
Baltimore, MD 21250
stephmilani@umbc.edu

Nicholay Topin
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
ntopin@cs.cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
katia@cs.cmu.edu

Abstract

In recent years, people have welcomed intelligent, autonomous agents into their homes and factories to perform various useful tasks. We will increasingly rely on these agents to assist with and make important decisions in scenarios that can be represented as sequential decision-making problems. In some of these problems, potential social ramifications and trade-offs must be considered. In these situations, it is essential for these agents to integrate human norms with traditional methods for learning in complex environments, such as reinforcement learning. In this work, we propose a novel framework, called Penalty-Modified Markov Decision Processes, for reinforcement learning in environments with potentially many norms. We formalize the learning and decision-making problem as solving a Markov decision process that is modified only as norms are violated. We show that the upper bound on the number of states created using our method is equivalent to the lower bound on the number of states created using existing approaches.

Keywords: norms, reinforcement learning, MDP, sequential decision making

Acknowledgements

Stephanie would like to thank the National Science Foundation Robotics Institute REU for funding this work.

*Work done as an intern at Carnegie Mellon University.

1 Introduction

Emerging autonomous agents will make increasingly more decisions that significantly impact human lives. These agents will reason about and take action in social environments, in which there are many socially-constructed norms, culturally- and socially-influenced preferences, and individual and collective values dependent on context and situation. To act appropriately in environments where actions may have social ramifications, these agents must integrate human values, norms, and preferences with traditional task-learning methods, such as reinforcement learning (RL). Integrating all of the aforementioned components is a lofty goal, as it requires the agent to reason about interconnected pieces at different levels of abstraction. Because norms are reflective of cultural, societal, and individual values and have an accompanying grounded action prescription, they are less abstract than values (e.g., the norm "always tell the truth" instead of the value "honesty") and can reflect both individual and societal preferences. This expressivity makes focusing on norms more desirable than focusing on values or individual preferences alone — especially in social environments.

Existing methods for incorporating normative reasoning into sequential decision-making problems suffer from a lack of scalability. In part, this issue stems from the coupling of norms and their resulting penalties: norms with shared penalties are considered to be separate, independent norms. However, penalties are often shared across various norm violations. Consider the penalties that arise from moving violations. Some of these penalties, such as fines, are immediate, one-time penalties that differ greatly in value (e.g., \$75, \$100). In addition to fines, a common long-term penalty for moving violations is demerit points on one's driving record. These points may arise from a number of different violations, such as speeding, failing to comply with the seatbelt law, or reckless driving. One loses one's driving license after accumulating enough points — regardless of the norm violations that led to the accumulation of these points.

To incorporate the observation of shared long-term penalties across different norms and to address the issue of scalability, we propose a novel framework, called *Penalty-Modified Markov Decision Processes* (PMMDPs), for RL agents to learn appropriate behavior in social environments with norms. We formalize the problem as solving a Markov decision process that is modified when norms are violated. We show that the upper bound for the number of states created by our method is equivalent to the tight bound of the number of states created using previous methods.

2 Background and Related Work

Markov decision processes (MDPs) are the standard formalism for modeling sequential decision-making problems. An MDP is a five-tuple consisting of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a state transition probability function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ that defines the transition dynamics, a reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and a discount factor $\gamma \in [0, 1]$ that represents the importance of future rewards relative to immediate rewards. In general, the goal of a planning or reinforcement learning (RL) agent is to maximize the expected cumulative discounted reward of its *policy*, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, which is a probability distribution over state-action pairs. Typically, in RL problems, the agent lacks direct access to $\mathcal{T}$ and $\mathcal{R}$, while planning problems assume that an agent has precise knowledge of these components.

Prior work in norm-aware reinforcement learning (RL) uses reward shaping to incorporate human desires that are independent from the task-completion goals [7], or has the agent learn an ethical utility function that is a part of the hidden state of a partially observable Markov decision process [1]. These approaches can only be used for norms with reward penalties; they cannot be used for norms with long-term penalties, such as those that modify the state space and the transition function. This is important because some penalties, such as demerit points on one's license for moving violations, cannot be effectively captured by reward penalties alone.

An alternative way to incorporate human-specified restrictions is the framework of Constrained MDPs [2]. Under this formulation, a domain contains multiple objectives. The goal is to maximize one of these objectives subject to constraints on the other objectives. Though an existing MDP can be modified by adding addition constraints, Constrained MDPs only restrict certain actions or apply immediate penalties. They do not provide a mechanism for changing environment dynamics as a result of constraint violations, so they are suitable for a different set of applications than our framework.

Another work [5], which is employed in planning with normative constraints, expresses penalties (or sanctions) as modifications to the agent's capability function and transition function. The capability function $\mathcal{C} : \mathcal{S} \rightarrow \mathcal{A}$ is a component of the MDP that denotes the set of admissible actions for each state. This approach does not include reward function modifications and strictly enforces action constraints on the agent.

A similar approach to ours, called *Normative Markov Decision Processes* (NMDPs), considers the full set of norms in the state space [4]. However, as noted in [6], using this representation in an RL setting causes the problem to succumb to the *curse of dimensionality* [3], meaning the problem is computationally intractable with a large number of norms. The *Modular Normative Markov Decision Process* (MNMDP) framework seeks to mitigate the issue of scalability with NMDPs by constructing a separate MDP for each individual norm and for each set of interacting norms [6], where the state representation for each MDP consists of the original components of the state space in addition to the relevant norm(s). Interacting norms are defined as norms which are simultaneously relevant to the agent's decision and must be considered

together. When there are many interacting norms in the environment, this representation suffers from increased state-space size — especially when it is not known a priori which and how many norms will interact. In some cases, using the NMDP approach is preferable over the MNMDP approach because the total number of states created is less in the former than the latter, which we prove in Section VI.

3 Penalty-Modified Markov Decision Processes

We introduce a new method, Penalty-Modified Markov Decision Processes (PMMDPs), for norm-aware reinforcement learning. Our method is motivated by the insight that, in the real world, norms often share penalties. We base our novel method on the standard Markov decision process (MDP) framework. In our framework, each Penalty-Modified MDP is a seven-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \mathcal{N}, \mathcal{P} \rangle$ consisting of the common MDP components, a totally ordered set of norms $\mathcal{N}$, and a totally ordered set of penalties $\mathcal{P}$ that arise from violating norms in $\mathcal{N}$. The set of norms and the set of penalties are each ordered based on their precedence; members of these sets are applied based on the order.

We depart from previous literature in our representation of norms by decoupling the changes that arise when an agent violates the norm from the norm itself. We present an initial, simplified norm representation with few components to highlight the novelty of our contribution; however, we could easily extend our representation to include more components used in other approaches, such as the authority that issued the norm. We represent a norm n in a totally ordered set of norms $\mathcal{N}$ as $n \in \mathcal{N} = \langle C, \sigma \rangle$, where C is the violation condition and σ consists of the set of MDP modifications, or sanctions, that arise when n is violated. The violation condition C is a propositional function that determines whether a norm has been violated, defined as $C : s_t \times a \times s_{t+1} \rightarrow \{0, 1\}$. For example, consider the norm of yielding to a pedestrian in a crosswalk. To test whether this norm has been violated by a driving agent, the propositional function tests for the presence of a pedestrian and whether the agent chose to continue driving instead of stopping. If the agent did not yield to the pedestrian, the propositional function evaluates to 1, indicating that the norm has been violated; otherwise, it evaluates to 0, indicating that the norm has not been violated.

As a result of violating a norm, the corresponding sanctions in σ are applied to the MDP. There are two components, $\sigma = \langle \sigma_R, \sigma_p \rangle$, which are modifications to the reward function and the state, respectively. The reward function modifier is the immediate reward penalty for violating n. It is represented by $\sigma_R : r_t \rightarrow r_m$, where r_t is the original reward received for taking the action and r_m is the new reward received based on the norm violation. This is akin to an agent receiving a fine for speeding, or another moving violation: the reward that may result from reaching the destination more quickly is augmented by the penalty for exceeding the speed limit. The state modifier consists of a penalty flag added to the following state to indicate the ongoing application of a penalty due to a violation. Specifically, σ_p corresponds to a specific feature which is normally 0 but is set to 1 when the norm is violated. This is akin to an agent receiving a demerit point on its driving record for a moving violation. Unlike the one-time penalty of a fine, a demerit point is a long-term penalty that remains on the agent's record until another event causes it to be removed.

The penalty flags are used by the set of penalties $\mathcal{P}$ for restricting or modifying future agent behavior. For each σ_p penalty flag set by one or more norms, $\mathcal{P}$ contains a transition function modifier $\mathcal{T}_p$ which modifies the transition function by mapping each s_{t+1} to an alternate s_{t+1}. Effectively, $\mathcal{P}$ defines a sequence of $\mathcal{T}_p$ which are applied after the standard transition function $\mathcal{T}$, but each $\mathcal{T}_p$ only changes the state if σ_p is 1 in the state. By separating norms from penalties, several different norms which result in the same long-term sanctions can share a feature indicating that the sanction should be applied due to a norm violation.

As in previous work, the other components of the MDP must also be modified to accommodate norms. The state space $\mathcal{S}$ now includes all states in the original state space (with all norm violation features set to 0) and all reachable states with at least one non-zero norm violation feature. This is similar to the modification performed when using an MNMDP, except we do not restrict the number of norms which can be simultaneously violated. The reward function $\mathcal{R}$ and transition function $\mathcal{T}$ are modified to be invariant to the norm violation features with the exception that $\mathcal{T}$ preserves norm violation flags (i.e., $\mathbb{P}(s_{t+1}|s_t, a_t) = 0$ if $s_t[\sigma_p] \neq s_{t+1}[\sigma_p]$ for any σ_p). The action space $\mathcal{A}$ and discount factor γ remain unchanged.

4 Learning in an Environment with Norms

Using the PMMDP framework, an agent can learn how to perform desired tasks in an environment with norms. Notably, the PMMDP can either be pre-computed as a pre-processing step or modified during the learning process. In this paper, we focus only on the modification of the PMMDP during the learning process; however, we plan to empirically demonstrate the efficacy of both approaches as future work.

The application of norm penalties to the MDP during the learning process proceeds as shown in Algorithm 1. When an agent takes an action in an environment, the original transition is computed without norms. Then, the transition is updated to account for the ongoing effects of previous norm violations, if any exist. The transition update is accomplished by sequentially applying the transition function modifiers, which correspond to the penalties of the violated norms. After

Algorithm 1 Modifying an MDP with Norm-Violation Penalties

function APPLY-PENALTIES(s_t, a, r_t, s_{t+1})
  for $\langle C, \langle \sigma_R, \sigma_p \rangle \rangle$ in $\mathcal{N}$ **do** ▷ sequentially check if norms are violated and apply corresponding sanctions
    if IS-IN-VIOLATION(C, s_t, a, s_{t+1}) **then** ▷ use condition function to check if agent violated norm
      $s_{t+1}[\sigma_p] \leftarrow 1$
      $r_t \leftarrow \sigma_R(r_t)$
  return s_{t+1}, r_t
function PERFORM-STEP-WITH-PMMDP(s_t, a)
  $s_{t+1}, r_t \leftarrow$ COMPUTE-TRANSITION(s_t, a) ▷ compute transition for unmodified state
  for $\langle \sigma_p, \mathcal{T}_p \rangle$ in $\mathcal{P}$ **do** ▷ sequentially apply transition function modifiers for norm violations
    if $s_{t+1}[\sigma_p]$ is 1 **then**
      $s_{t+1} \leftarrow \mathcal{T}_p(s_{t+1})$
  $s_{t+1}, r_t \leftarrow$ APPLY-PENALTIES(s_t, a, r_t, s_{t+1})
  return s_{t+1}, r_t

that, the transition is evaluated for norm violations. For each norm $n \in \mathcal{N}$, the corresponding propositional function is evaluated. If the agent violated the norm in question, then σ_R is applied to the reward for this transition and σ_p is applied to the following state. The agent then transitions to the new state s_{t+1} that includes the new penalty modification and receives the modified reward.

Consider an agent in a realistic driving domain. When the agent takes an action, the transition is computed without considering traffic laws. Then, the transition is updated to account for the ongoing effects of previous moving violations (if they exist), such as points on its driving record. After that, each relevant traffic norm (e.g., speed restrictions, turn-signal regulations) is examined to determine if any of these norms were violated by agent. Any fines (reward penalties) are then applied to the reward for this transition and any demerit points (penalty flags) are then applied to the following state. The agent then transitions to the new, modified state that includes the demerit point(s) and receives the modified reward that includes the fine(s). For example, if an agent did not yield to a pedestrian in a crosswalk, it would receive a fine of $\$100$ and a demerit point on its driving record. The modified reward would include this fine and the modified state would include this demerit point. The demerit point will affect future transitions through the corresponding penalty-specific transition function that will be applied after the standard transition function.

5 Analysis

In this section, we show an upper bound on the number of states in a PMMDP that is of the same order as the lower bound on the number of states created using existing methods. This result shows that our formulation of norms scales at least as well as existing methods, even in the worst case. We first demonstrate that there exist some cases where using NMDPs is preferable to using MNMDPs, which has not previously been demonstrated. Then, we show that our method is always at least as good as the MNMDP and NMDP approaches with respect to the number of states created. Notably, with our approach, the state-space complexity of the MDP is only increased with the number of penalties that are imposed by the norm violations. If the agent does not violate any norms, then no penalties are imposed, and, thus, the agent solves the original MDP, making the best-case size of the state space equivalent to that of the original, normless MDP. Importantly, this analysis is only for binary norm violation features; future work will extend this method to handle non-binary features. See Table 1 for an overview of our analysis.

Theorem 1. *There exist some cases where the NMDP formulation has fewer states than the MNMDP formulation.*

Proof. Let n be the total number of norms and d be the total number of possible interactions between norms, where an interaction is defined as two or more norms that are concurrently activated in a scenario. As shown by the authors when the MNMDP approach was introduced [6], the number of states in the fully-normative MDP framework is of the order $\Omega(2^n)$ and the number of states in the MNMDP framework is of the order $\Omega(n^d)$. That means that using the MNMDP framework is preferable if $n^d < 2^n$, which can be converted to $\log_2(n^d) < n$. Then, $d(\log_2 n) < n$, so $d < n/\log_2 n$. Thus, the MNMDP framework is preferable to the fully normative MDP approach when the number of interactions, or concurrently active norms, is less than $n/\log_2 n$.

Theorem 2. *The number of states in a PMMDP is never more than the number of states in the corresponding MNMDP nor the number of states in the corresponding NMDP.*

Approach	Number of States Created	Note
NMDP	$\Omega(2^n)$	
MNMDP	$\Omega(n^d)$	If $d > n/\log_2(n)$, more states created than NMDP
PMMDP	$\mathcal{O}(2^p), \mathcal{O}(p^d)$	$p \leqslant n$

Table 1: Analysis of the number of states created for each of the three methods. The state-space size for NMDPs is exponential in n, where n is the number of norms that can be either on or off. The state-space size for MNMDPs is exponential in d, where d is the maximum number of interactions between norms. The worst-case state-space size for PMMDPs is exponential in p or exponential in d; importantly, $p \leqslant n$, so no more states are ever created than in the other two methods.

Proof. Let p be the number of possible penalties. In the worst case of a one-to-one mapping of penalties to norms, where each norm has its own unique penalty, the number of penalties p is equivalent to the number of norms n ($p = n$); however, in general, $p \leqslant n$ because the set of norms to penalties consists of one-to-one or many-to-one mappings. Let $|P|$ be the number of possible concurrent penalties. Because $p \leqslant n$, $|P| \leqslant 2^p \leqslant 2^n$. Continuing with the aforementioned notation in the proof of Theorem 1, let d be the maximum number of concurrent norms. Then,

$$|P| = \sum_{i=1}^{d} \binom{p}{d} = \binom{p}{1} + ... + \binom{p}{d} \rightarrow p^1 + ... + p^d \tag{1}$$

Hence, the number of states created in our approach is $O(p^d)$, meaning that the upper bound of the number of states created in our approach is equivalent to the lower bound on the number of states created by the NMDP and MNMDP methods.

6 Conclusion

We present a novel framework for RL with normative constraints. The number of states created in our framework is less than other frameworks because it depends on the number of penalty flags, not on the number of norms ($p \leqslant n$). The computed worst-case bound on the number of states created by our method is of the same order as the lower-bounds for previous work. Our framework also avoids redundant states, adding no more states than are included in the naive, fully normative case. Furthermore, our framework is the first of its kind in an RL setting that includes penalties for norm violations that modify the reward function, the transition function, and the state space.

References

[1] D. Abel, J. MacGlashan, and M. L. Littman. Reinforcement learning as a framework for ethical decision making. In *AAAI Workshop on AI, Ethics, and Society*, 2016.

[2] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[3] R. E. Bellman and S. E. Dreyfus. *Applied dynamic programming*. Princeton University Press, 4th edition, 1971.

[4] M. S. Fagundes, S. Ossowski, J. Cerquides, and P. Noriega. Design and evaluation of norm-aware agents based on normative markov decision processes. In *International Journal on Approximate Reasoning*, volume 78, 2016.

[5] M. S. Fagundes, S. Ossowski, M. Luck, and S. Miles. Using normative markov decison processes for evaluating electronic contracts. In *AI Communications*, volume 25, 2012.

[6] V. Krishnamoorthy, W. Luo, M. Lewis, and K. Sycara. A computational framework for integrating task planning and norm aware reasoning for social robots. In *RO-MAN*, 2018.

[7] Y.-H. Wu and S.-D. Lin. A low-cost ethics shaping approach for designing reinforcement learning agents. In *AAAI*, 2018.

Inferring Value by Coherency Maximization of Choices and Preferences

Adam N. Hornsby
Department of Experimental Psychology
University College London
London
adam.hornsby.10@ucl.ac.uk

Bradley C. Love
The Alan Turing Institute
United Kingdom
b.love@ucl.ac.uk

Abstract

In standard models of reinforcement learning (RL), the reward signal is objective and drives learning. For example, in a video game, the points earned serve as an accessible and objective measure of reward that can be maximized by an agent. However, outside the confines of such artificial environments, rewards are not specified. For example, no objective rewards are associated with choosing to eat a pizza or spending time with a friend. In such cases, which encompass almost all of human experience, the subjective value of the choice is interpreted by the agent by comparing how well the choice aligns with the agent's preferences. The agent can then update its preferences in the absence of an objective reward, which in turn may alter future valuations of choices. To date, few RL models have formalized this process of subjective reinforcement learning. We propose a new computational cognitive model which characterizes how people make subjective decisions and update their preferences over time. The probability of a choice is a determined by how similar choice options (e.g., pizza) are to the agent's preference vector, where similarity is a function of attention-weighted distance such that some attributes (e.g., taste) can be weighted more than others (e.g., calories). Preferences are updated by gradient-descent learning rules that make repeating related choices (e.g., pizza over salad) more likely in the future by adjusting attention weights and the position of the preference vector. These learning rules maximize coherency by aligning preferences to match choices, a well-documented finding within the psychological literature of free choice. This model, which radically departs from standard RL models, is validated by simulation and behavioral experiments with humans. People updated their preferences and generalized to similar choices in a manner consistent with the model.

Keywords: Subjectivity, Reinforcement Learning, Preferences, Cognitive Modelling, Intrinsic Motivation, Coherency Maximization

Acknowledgements

A.N.H is supported by dunnhumby and the Royal Commission for the Exhibition of 1851. B.C.L. is supported by a Wellcome Trust Senior Investigator Award WT106931MA, and National Institute of Child Health and Human Development Grant 1P01HD080679

1 Introduction

Many standard models of reinforcement learning (RL) assume that rewards are the primary source of learning and that agents are motivated to maximize them [1, 2]. It is therefore common to train RL agents in extrinsically rewarding environments, where they can learn the correct policy by adapting to feedback from the environment. For example, artificial agents have been shown to reach superhuman levels of play in games, where rewards are received for effective play [3]. Those interested in human behaviour have demonstrated similarities between learning in artificial agents and people in such tasks [2]. For example, evidence of a neural substrate for reward prediction error suggests that the brain keeps track of extrinsic rewards associated with actions in a way similar to a popular RL algorithm, known as TD-learning [4]. This has led to much excitement about the similarities between RL models and humans.

Despite these successes, many decisions in real life are made without an obvious extrinsic reward. For example, people must choose between products in the supermarket and between presidential candidates when voting. Indeed, when making such choices, people may feel strongly about choices that have little extrinsic advantage whatsoever. For example, while cheaper instant coffee may be lighter on the wallet, it may taste worse than the competitors. This trade-off may or may not be palatable, depending on people's preferences for price and taste. In these cases, the subjective value of a choice is determined by integrating over a range of different reward signals in relation to one's preferences. The result is a single measure of subjective value; a phenomenon that is predicted by dopamine signals in the brain [5, 6]. Yet — due to use of extrinsically rewarding tasks — few models have formalized the process by which subjective preferences may be learned and how they may interact with decisions over time.

To understand how people learn subjective preferences, it is necessary to consider people's intrinsic motivations. Most primitively, species are motivated to preserve homeostasis; seeking food when they're hungry and rest when they're tired. Thus, some have proposed that intrinsic reward functions exist to motivate self-regulation [7, 8]. However, this motivation may not fully account for all subjective decision making. For example, a large part of learning in childhood is driven by play, suggesting that people are also driven by higher-level drives, such as curiosity [9]. Indeed, advances have shown artificial curiosity to elicit good performance in sparsely-rewarding games [10, 11]. While this is promising, there are likely other cognitive drives that motivate learning of subjective preferences beyond what has been explored here.

In addition to being curious, people are also motivated to appear coherent with their past choices. For example, a recent analysis of 280,000 real British consumers' showed that they had a reduced tendency to explore new products the more they repeat-purchased (i.e., exploited) the same one [12]. This is surprising, in that it is the opposite of what would be expected of an agent motivated to minimize uncertainty in the environment. Rather, it suggests that people are motivated to maximize the coherency between their preferences and past choices, meaning that they come to prefer the things they choose. While simple, this kind of internal consistency is pivotal to many rational models of decision making [13]. For example, it is often assumed that choices should be stochastically transitive [5]. Thus, in the absence of an extrinsic reward signal, coherency may be the most rational strategy a person can fall back on. The notion that choices may be self-reinforcing is not something that has been widely explored within RL and is therefore in need of investigation.

In this research, we present a new cognitive computational model of human preference formation and subjective decision making. Inspired by RL, the model makes decisions and learns from these to update its preferences. However, unlike standard models of RL, it does not assume that a reward must exist in order for an agent to learn. Specifically, the agent makes choices and uses those as a basis to update their preferences and attentional focus. Thus, we say that the model is motivated to *maximize coherency* between one's past and present choices. Uniquely, the model learns preferences over attributes of choices rather than choices themselves. This makes unique predictions about multi-attribute generalization (or "spillover" effects), where people increase their preference towards attributes that discriminate in the choice just made; a result predicted by "blocking effects" in associative learning [14]. In the remainder of this article, we introduce the cognitive model and results from a laboratory experiment that confirms the existence of spillover effects caused by coherency maximization and thus several key tenets of the model.

1.1 Model

We now describe our model of subjective preference learning and decision making. Note that vectors will now be denoted in **bold** lowercase letters and matrices in **bold** uppercase letters.

Broadly speaking, the model works by maintaining an internal set of preferences and attention weights for attributes across choices. For example, products in a supermarket can be described in terms of their nutritional content or whether they are found in a salad [15]. Individuals will possess different preferences for those attributes, and pay differing levels of attention to them. Preferences are therefore represented as ideal-points within a multidimensional space and — alongside attention weights — used to determine how favourable a choice is at a given timepoint. In particular, the higher the attention-weighted similarity, the more likely it will be to choose that option. Much like a reinforcement

learning agent, the model takes an action, observes its environment, updates its internal state and then repeats the process.

We denote the observation of options in the environment using the matrix **O**, which has a shape of $N \times M$. Here, N denotes the number of choices available $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_N]^T$ at a given timestep. For simplicity of notation, we assume that the model must choose between two items at any one time (i.e. N=2). However — in principle — the model is not constrained to this. M denotes the number of attributes for each option. Thus, each column $\mathbf{o}_i$ ($i \in \{1, ...N\}$) is a vector of M attributes $\mathbf{o}_i = [o_{i1}, o_{i2}, ..., o_{iM}]$. Therefore, the element o_{ij} corresponds to the jth attribute ($j \in \{1, ...M\}$) of the ith item.

Action selection

In order to determine the most appropriate choice, the model first calculates a probability over the available options observed in **O** using the preference vector $\mathbf{p} = [p_1, p_2, ..., p_M]^T$ and the attention weight vector $\mathbf{w} = [w_1, w_2, ..., w_M]^T$. Each element of the preference and attention weight vectors p_j and w_j maps to an attribute j in the attribute vector o_{ij}.

In order to determine the probability of an action, the model calculates an attention-weighted similarity between the preference vector and each of the N = 2 item vectors $\mathbf{o}_i$ within the observation matrix **O**. We denote the attention-weighted similarity as $a(\mathbf{o}_i)$

$$a(\mathbf{o}_i) = -\gamma \left(\sum_{j=1}^{M} w_j (o_{ij} - p_j)^2 \right)^{1/2} \tag{1}$$

Where γ is a scaling hyperparameter. Note that this weighted Euclidean similarity term is very similar to the one used in the ALCOVE model of human categorization [16].

In order to determine the probability of selecting an option i, the attention-weighted similarity $a(\mathbf{o}_i)$ is then fed into a softmax function

$$\mathbf{f}(\mathbf{o}_i) = \mathbb{P}(I_i|\mathbf{O}) = softmax(\mathbf{a}(\mathbf{O}))_i = \frac{\exp a(\mathbf{o}_i)}{\sum_{k=1}^{N} \exp a(\mathbf{o}_k)} \tag{2}$$

Action selection

Choices can be selected using one of the many popular strategies used in RL, such as ϵ-greedy or softmax action selection [1]. In each case, higher probabilities for choices (i.e. stronger preferences) increase the likelihood of exploiting that known favourite, rather than exploring disfavoured options. When using softmax selection specifically, the λ parameter can be thought of as determining the "fussiness" of the agent's choices, such that higher λ equates to a higher likelihood of choosing the favorite. We denote the choice made by the agent as c.

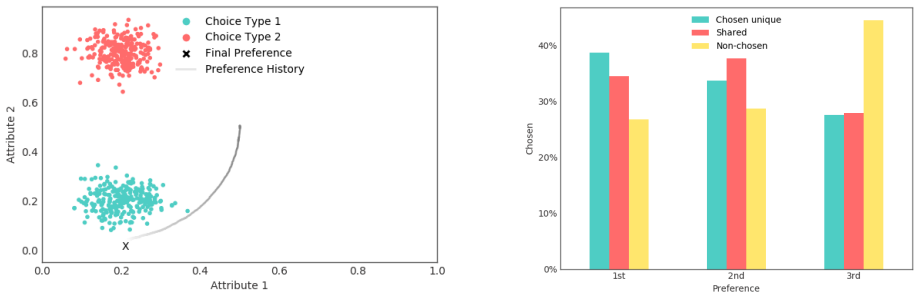
Updating preference and attention-weight vectors

Following an action, the agent must then update its preference and attention weight vectors. As previously discussed, people have a strong desire to make subjective choices in a way that is internally consistent. Data of consumer behaviour has shown that — in subjective choice domains where there is no explicit feedback — preferences tend to follow choices [12]. This has also been demonstrated inside the lab. Specifically, psychological studies of "free choice" have shown that making a forced choice between items causes someone to increase their preference for it; a process that appears to be reinforced by activity in the basal ganglia [17, 18]. We therefore update the preference and attention weight vectors so as to maximize the likelihood of the previous choice.

The exact learning procedure used to update the preference and attention weight vectors is gradient descent on the cross-entropy loss. After selecting an action, the cross-entropy loss is calculated between the action probabilities output by the model $\mathbf{f}(\mathbf{o}_i)$ and the actual choice c that was made.

$$l(\mathbf{f}(\mathbf{O}), c) = -\sum_{i=1}^{N} \mathbb{1}_{\{c=i\}} log(f(\mathbf{o}_i)) \tag{3}$$

After making an action, the preference and attention weights are updated so as to minimize the cross-entropy error. Concretely, they are updated proportionally to the negative of the error gradient (i.e., gradient descent). For brevity, the derivatives are not reported here.



(a) (b)

Figure 1: Figure (a) shows results from the simulation, in which the agent is requested to choose between choice type 1 and 2 over 10,000 timeteps. The gray preference history is coloured by the ratio of the attention weights[1], where lighter values indicate a higher attention paid towards the second attribute. Figure (b) shows results from the experiment, demonstrating the proportion that each pattern was chosen as a first, second and third preference.

2 Results and discussion

Model simulation

To demonstrate how this computational model behaves, we performed a simulation in a simple two-dimensional environment, in which there are two choices that don't vary on a first dimension but vary significantly on a second dimension. This is analogous to choosing between two cola brands that vary little in taste but differ distinctly in the color of the branding. The results of this simulation are plotted in Figure 1a. After simulating 10,000 forced-choices using ϵ-greedy action selection, one can see that the model eventually — due to happenstance — comes to prefer items with attributes resembling those of choice type 1. Perhaps most interestingly, as it moves towards this choice type, the preferences and attention of the model moves most in favour of the attributes of the choice that make it appear unique. Thus, the simulation highlights two predictions made by our model. Firstly, it predicts that people will generalize their preferences to novel items if they share attributes with items that they have already chosen. Secondly, people should overemphasize their preference for attributes that made their choice unique. For example, a person should have an exaggerated preference for other brands that use the colour unique to their preferred cola brand, even when they have never tried them. In the remainder of this paper, we report results from a controlled experiment that attempts to test these key predictions.

Coherency maximization generalizes to similar options

To test the two aforementioned predictions of this model, 1003 participants were recruited to an online study via Amazon Mechanical Turk. Participants were asked to design a robot. They were then introduced to a second robot, before both turned around revealing previously unseen, randomly assigned patterns on their backs. Finally, participants were asked to choose between three patterns; one that was unique to the back of the robot they had previously chosen (i.e. chosen unique), another that was shared across the backs of the two robots and a final pattern that was unique to the back of the robot they hadn't designed (i.e. non-chosen). Firstly, it was hypothesized that — in accordance with the model presented here — participants' would prefer novel patterns that were associated with the robots they had previously designed, implying that people reason about options and preferences within a multidimensional space. Secondly, it was hypothesized that participants would have an exaggerated preference for patterns unique to the previously designed robot; analogous to the well-documented "blocking effect" found in studies of associative learning [14]. To foreshadow, results from the experiment confirmed both of these hypotheses.

Results from the experiment are shown in Figure 1b. To assess the significance of the effects, the rank sums of the preferences for each choice type were computed for each participant. A non-parametric Friedman test of differences among repeated-measures was conducted on the rank sums rendered significant ($\chi^2 = 137.48, p < 0.001$), suggesting that there were significant differences among the preferences of participants for each choice type. To further assess the significance of the differences between each of the choice types, three non-parametric Wilcoxon signed-rank tests were conducted. Each p value was therefore compared to a Holm-Bonferroni corrected value. The first test comparing the summed preferences between the chosen-unique (Median = 9, IQR = 4.0) and shared items ($Median$ = 9, IQR = 3.0) was significant ($Z = -2.91, p < 0.005, r = 0.09$). The second test comparing the summed preferences between the chosen-unique and non-chosen (Median = 11, IQR = 5.0) items was also significant ($Z = -12.08, p < 0.001, r = 0.39$). The final test comparing the summed preferences between the non-chosen and shared items was also significant ($Z = -11.70, p < 0.001, r = 0.38$). These results therefore support our key hypotheses, in that they suggest making a

forced-choice between options increases liking for novel options that are *similar* items just chosen and that people will overemphasize their preference for attributes that made their choice unique.

In typical studies of RL, it is assumed that agents are motivated to maximize extrinsic reward and that this reward forms the basis of learning. Yet, in real life, people often make decisions for which there is no clear reward signal. How can people learn from such decisions? In this research, we argue that people have a fundamental desire to maximize coherency between their past choices and present preferences. Uniquely, we extend existing results in the context of "free choice" by showing that — as people reason about options in terms of a multi-attribute space — the desire to maximize coherency can cause spillover effects, in which people come to prefer novel options that are similar (i.e., share attributes) to ones previously chosen [17, 18]. In addition, the results show that — analogous to cue-competition effects in associative learning [14] — people place an additional emphasis on the attributes of the choice that make it unique.

While in its infancy, this research hints at several challenges to the application of RL to subjective decisions. In particular, it suggests that learning and decision making are fundamentally different depending on the presence of extrinsic reinforcement. For subjective decisions, people may not require reward to learn at all. Instead, it is possible that they simply use their past choices to infer and update their preferences. Consequently, this can materialize in behaviour that wouldn't otherwise be predicted by uncertainty-minimizing agents, such as a reduced tendency to explore new options the more they exploit [12]. Indeed, in this model, there are no "pure explorations", as all choices exist within a known, multi-dimensional space. In future research, we aim to evaluate the claim that uncertainty minimizing and coherency maximizing can be elicited by rewards or the lack thereof, respectively. What is clear is that a new theory of subjective RL may be required before these models can account for an important category of decisions made in the wild.

References

[1] R. Sutton and A. Barto, "Reinforcement Learning: An Introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1054–1054, 1998.

[2] W. Schultz, "Behavioral Theories and the Neurophysiology of Reward," *Annual Review of Psychology*, vol. 57, no. 1, pp. 87–115, 2006.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[4] J. P. O'Doherty, P. Dayan, K. Friston, H. Critchley, and R. J. Dolan, "Temporal difference models and reward-related learning in the human brain," *Neuron*, vol. 38, no. 2, pp. 329 – 337, 2003.

[5] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, 1997.

[6] A. Lak, W. R. Stauffer, and W. Schultz, "Dopamine prediction error responses integrate subjective value from different reward dimensions," *Proceedings of the National Academy of Sciences*, vol. 111, no. 6, pp. 2343–2348, 2014.

[7] S. P. Singh, R. L. Lewis, A. G. Barto, and J. Sorg, "Intrinsically motivated reinforcement learning: An evolutionary perspective," *IEEE Transactions on Autonomous Mental Development*, vol. 2, pp. 70–82, 2010.

[8] M. Keramati and B. Gutkin, "Homeostatic reinforcement learning for integrating reward collection and physiological stability," *eLife*, vol. 3, p. e04811, dec 2014.

[9] P.-Y. Oudeyer, "Computational theories of curiosity-driven learning," 2018.

[10] J. Schmidhuber, "Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity creativity," 2007.

[11] Y. Burda, H. Edwards, D. Pathak, A. J. Storkey, T. Darrell, and A. A. Efros, "Large-Scale Study of Curiosity-Driven Learning," *CoRR*, vol. abs/1808.04355, 2018.

[12] P. S. Riefer, R. Prior, N. Blair, G. Pavey, and B. C. Love, "Coherency-maximizing exploration in the supermarket," *Nature Human Behaviour*, 2017.

[13] L. J. Savage, *The Foundations of Statistics*. Wiley Publications in Statistics, 1954.

[14] L. Kamin, "Predictability, surprise, attention and conditioning. in ba campbell & rm church (eds.), punishment and aversive behavior (pp. 279-296)," *New York: Appleton-Century-Crofts*, 1969.

[15] A. N. Hornsby, T. Evans, P. Riefer, R. Prior, and B. C. Love, "Conceptual organization is revealed by consumer activity patterns," 2018.

[16] J. K. Kruschke, "ALCOVE: An exemplar-based connectionist model of category learning," *Psychological Review*, vol. 99, pp. 22–44, 1992.

[17] J. W. Brehm, "Postdecision changes in the desirability of alternatives," *Journal of Abnormal and Social Psychology*, 1956.

[18] J. Cockburn, A. G. E. Collins, and M. J. Frank, "A Reinforcement Learning Mechanism Responsible for the Valuation of Free Choice," *Neuron*, vol. 83, no. 3, pp. 551–557, 2014.

Learning Curriculum Policies for Reinforcement Learning*

Sanmit Narvekar
Department of Computer Science
University of Texas at Austin
sanmit@cs.utexas.edu

Peter Stone
Department of Computer Science
University of Texas at Austin
pstone@cs.utexas.edu

Abstract

Curriculum learning in reinforcement learning is a training methodology that seeks to speed up learning of a difficult target task, by first training on a series of simpler tasks and transferring the knowledge acquired to the target task. Automatically choosing a sequence of such tasks (i.e., a *curriculum*) is an open problem that has been the subject of much recent work in this area. In this paper, we build upon a recent method for curriculum design, which formulates the curriculum sequencing problem as a Markov Decision Process. We extend this model to handle multiple transfer learning algorithms, and show for the first time that a *curriculum policy* over this MDP can be learned from experience. We explore various representations that make this possible, and evaluate our approach by learning curriculum policies for multiple agents in two different domains. The results show that our method produces curricula that can train agents to perform on a target task as fast or faster than existing methods.

Keywords: Reinforcement Learning; Transfer Learning; Curriculum Learning

Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-1637736, IIS-1651089, IIS-1724157), Intel, Raytheon, and Lockheed Martin. Peter Stone serves on the Board of Directors of Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

*A full-length, complete version of this paper will appear at AAMAS 2019, and can be found at https://arxiv.org/abs/1812.00285

1 Introduction

Over the past two decades, transfer learning [5, 11] is one of several lines of research that have sought to increase the efficiency of training reinforcement learning agents. In transfer learning, agents train on simple *source* tasks, and transfer knowledge acquired to improve learning on a more difficult *target* task. Typically, this has been a one-shot process, where information is transferred from one or more sources directly to the target task. However, as the problems we task reinforcement learning agents with become ever more complex, it may be beneficial (and even necessary) to gradually acquire skills over multiple tasks *in sequence*, where each subsequent task builds upon knowledge gained in a previous task. This insight is the basis for *curriculum learning* [1, 6].

The goal of curriculum learning is to design a sequence of source tasks (i.e. a curriculum) for an agent to train on, such that after training on that sequence, learning speed or performance on a *target* task is improved. Automatically designing a curriculum is an open problem that has only recently begun to be examined [4, 3, 8, 2, 7, 10]. One recent approach [7] proposed formulating the selection of tasks using a (meta-level) curriculum Markov Decision Process (MDP). A policy over this MDP, called a curriculum policy, maps from the current knowledge of an RL agent to the task it should learn next. However they did not demonstrate whether the curriculum policy could actually be learned. Instead, they proposed an algorithm to approximate a single execution of the curriculum policy, corresponding to an individual curriculum.

Until now, it was not known if curriculum policies could be learned: that is, whether it is possible to find a representation that is both compact enough and generalizable enough to facilitate learning. Our main contribution is to demonstrate that curriculum policies can indeed be learned, and we explore various representations that make this possible. In addition, we generalize the curriculum MDP model proposed by Narvekar et al. [7] to handle different kinds of transfer learning algorithms. Finally, we empirically show that the curricula produced by our method are at least as good as, or better than those produced by two existing curriculum methods on two different domains. We also demonstrate that curriculum policies can be learned for agents with different state and action spaces, agents that use different transfer learning algorithms, and different representations for the curriculum MDP.

2 Learning Curriculum Policies

Our work extends the model proposed by Narvekar et al. [7], which formulates curriculum generation as an interaction between two separate Markov Decision Processes: one is for a *learning agent* that is trying to solve a specific target task MDP M_t, as is the standard case in reinforcement learning. The second is a *curriculum agent*, which interacts in a second, higher level *curriculum MDP* (CMDP), and whose goal is to sequence tasks M for the learning agent. A curriculum MDP was defined to be an MDP where: (1) the state space $\mathcal{S}^C$ consists of all policies the learning agent can represent; (2) the action space $\mathcal{A}^C$ is the set of tasks to train on; (3) the transition function p^C reflects the change in the learning agent's policy as a result of learning a task; and (4) the reward r^C is the cost in time to learn the task. The starting state corresponds to a random learning agent policy, and terminal states are learning agent policies that can achieve a predefined performance level on a target task.

One limitation of the previous definition is that it assumes the underlying transfer learning mechanism is value function or policy transfer. Intuitively, the state space of a CMDP should represent different states of knowledge. The goal of the agent is to reach a state of knowledge that allows solving the target task in the least amount of time. As a case study, in this paper we consider learning agents that use two different transfer algorithms: value function transfer and potential-based reward shaping. In value function transfer, the value function learned in a source task is used to initialize the value function in a subsequent task. In potential-based reward shaping, the value functions of a set of source tasks are used as potential functions to create a shaping reward for a next task (see Svetlik et al. [10] for details). Thus, for an agent that uses reward shaping, the CMDP state is represented as a set of potential functions, and the goal is to find a state whose potentials allow learning the target task as fast as possible.

2.1 Representing CMDP State Space

In the standard reinforcement learning setting, the agent perceives its state as a set of state variables. These are typically used to extract basis features $\boldsymbol{\phi}(\mathbf{s})$, which transform the state variables into a space more suitable for learning and use in function approximation. Given these features and a functional form, the goal is to learn weights $\boldsymbol{\theta}$ for the value function or policy. We introduce an analagous process for curriculum design agents acting in CMDPs. We will ground the discussion assuming both the learning and curriculum agents use a value-function-transfer-based approach. However, the idea is easily applied to a reward-shaping setting by noting that the reward can also be expressed as a product of state features and weights $r(s,a) = \boldsymbol{\phi}(s,a) \cdot \boldsymbol{\theta}$.

The first question is how to represent the raw state variables of a CMDP state. The representation chosen must be able to represent *any* policy the underlying learning agent can represent (or equivalently, any shaping reward). Assuming the learning agent derives its policy from an action-value function $Q_{\boldsymbol{\theta}}(s,a)$, the form of the function (such as network

architecture, etc.) determines the class of policies that can be represented. This functional form $Q_{\boldsymbol{\theta}}(s,a)$ and how learning agent features $\boldsymbol{\phi}$ are extracted are fixed. Thus, it is specific values of the weight vector $\boldsymbol{\theta}$ that actually instantiates a policy in this class. Therefore, it follows that we can represent the state variables for a particular CMDP state s^C using the instantiated vector of learning agent weights: $s^C = \boldsymbol{\theta}$. Different instantiations of $\boldsymbol{\theta}$ correspond to different CMDP states. Typically, these weights $\boldsymbol{\theta}$ will take on continuous values. Therefore, in order to learn a CMDP action-value function $Q^C_{\boldsymbol{\theta}^C}(s^C, a^C)$, it will be necessary to do some kind of function approximation.

First we consider one way of extracting CMDP state features and performing function approximation, when the domain has a finite state space. Assume again the learning agent learns an action-value function $Q_{\boldsymbol{\theta}}(s,a)$, for each state-action pair in the task. We can represent Q as a linear function of "one-hot" features $\boldsymbol{\phi}(s,a)$ and their associated weights $\boldsymbol{\theta}$ as $Q_{\boldsymbol{\theta}}(s,a) = \boldsymbol{\theta} \cdot \boldsymbol{\phi}(s,a)$. In other words, all the action-values are stored in $\boldsymbol{\theta}$, and $\boldsymbol{\phi}(s,a)$ is a one-hot vector used to select the activated action-value from $\boldsymbol{\theta}$. One approach for designing $\boldsymbol{\phi}^C$ is to utilize tile coding [9] over subsets of action-values in $\boldsymbol{\theta}$. Specifically, the idea is to create a separate tiling for each primitive state s in the domain. Each such tiling will be defined over the action-values in $\boldsymbol{\theta}$ associated with state s. Thus, this creates $|\mathcal{S}|$ tiling groups, where each group is defined over $|\mathcal{A}|$ CMDP state variables (i.e. action-values). To create the feature space, multiple overlapping tilings are laid over each group.

The representation problem is harder in the continuous case, since each parameter θ_i is not local to a state, and we cannot use a state-by-state approach to create a basis feature space. In principle, any continuous feature extraction and function approximation scheme can form the basis of $\boldsymbol{\phi}^C$ (tile coding, neural nets, etc.), and would need to be tailored to the domain.

3 Experiments

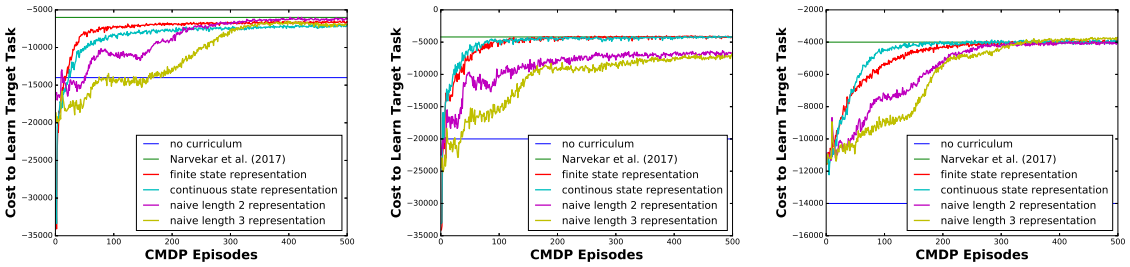
We evaluated learning curriculum policies for agents on a grid world domain used in previous work [7] as well as a Ms. Pac-Man domain. We will show the results as CMDP learning curves. The x-axis on these learning curves are over *CMDP episodes*. Each CMDP episode represents an execution of the current curriculum policy for the agent. Thus, multiple tasks are selected over the course of a single episode, with each task taking a varying number of steps/episodes, which contributes to the cost on the y-axis. Tasks are selected until the desired performance can be achieved in the target task, at which point the CMDP episode is terminated. In short, the curves show how long it would take to achieve a certain performance threshold on the target task following a curriculum, where the curriculum is represented by the CMDP policy, which is being learned over time.

We compare curriculum policies learned for each agent to two static curricula. The first is the baseline *no curriculum* policy. In this case, on each episode, the agent learns tabula rasa directly on the target task. The flat line plotted represents the average time needed to directly learn the target task. Note that the line is flat because the curriculum is fixed and does not change over time. The second is a curriculum produced by following an existing curriculum algorithm ([7] for the gridworld, [10] for Ms. Pac-Man, to compare with past work). We also compare to a naive learning-based approach, which represents CMDP states using a list of all tasks learned by the learning agent. For example, the start state is the empty list. Upon learning a task M_1, the CMDP agent transitions to a new state $[M_1]$. In order to deal with the combinatorial explosion of the size of the state space, we limit the number of tasks that can be used as sources in the curriculum to a constant (between 1 and 3 in our experiments), and force the selection of the target task after.

3.1 Gridworld Experiments

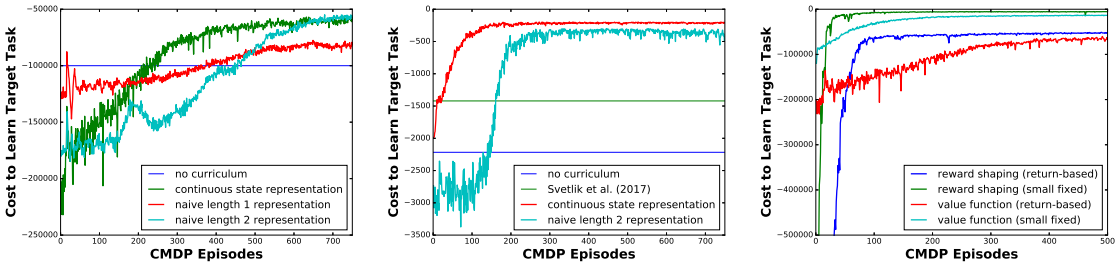
In this experiment, we examine learning curriculum policies for 3 learning agents that have different state and action spaces (see [7] for learning agent details), but use the same transfer learning algorithm (value function transfer), in a simple grid world domain. In particular, we examine and compare two different types of representations for the CMDP state. The first CMDP representation is based on the finite state space representation discussed earlier. The learning agents use Sarsa(λ) with an *egocentric* feature space. Thus, the parameters $\boldsymbol{\theta}$ learned are not action-values for each state. However, since the underlying domain has a fixed number of states, we can move the learning agent to each of the states in the target task and compute action values for each grid cell. Let this new parameter of weights be $\boldsymbol{\theta'}$. We can now utilize the procedure described in Section 2 to create a CMDP feature space $\boldsymbol{\phi}^C(\boldsymbol{\theta'})$. The second CMDP representation was created directly from $\boldsymbol{\theta}$ without using an intermediary state-based action-value representation. We did this by creating a separate tile group directly for each θ_i.

A total of 9 different tasks were created to form the action space $\mathcal{A}^C$ of the CMDP agent. Each of the learning agents was trained until it could receive a return of 700 on the target task. The CMDP learning curves for each agent are shown in Figures 1(a) - 1(c). The results show that each agent successfully learned curriculum policies using both CMDP representations that were comparable in performance to the curricula generated by previous work [7].



(a) (b) (c)

Figure 1: CMDP learning curves for the (a) basic agent, (b) action-dependent agent, and (c) rope agent using different curriculum design approaches and CMDP state space representations. All curves are averaged over 500 runs.



(a) (b) (c)

Figure 2: CMDP learning curves on the Ms. Pac-Man target task, using (a) value function transfer and (b) transfer with reward shaping. All curves are averaged over 500 runs. Cost is measured in game steps for (a), and episodes for (b).

3.2 Ms. Pac-Man Experiments

We also evaluated learning CMDP policies in the game of Ms. Pac-Man. In this experiment, we explore learning a curriculum policy for a Ms. Pac-Man agent, when the agent uses 2 different types of transfer learning methods: value function transfer and reward shaping. The learning agents were based off the agents used in previous work [10]

In the value function case, the raw CMDP state variables s^C are the weights $\boldsymbol{\theta}$ of the Ms. Pac-Man agent's linear function approximator. To create the CMDP space $\boldsymbol{\phi}^C$, we normalize $\boldsymbol{\theta}$ and use tile coding, creating a separate tiling over each θ_i. In the reward shaping setting, each source task in the curriculum is associated with a potential function (derived from the value function). As multiple tasks are learned, the potentials are added together, and used to create a shaping reward (as done in [10]). Thus, the raw CMDP state variables are the summed weights of the potential functions. As in the value function case, we use tile coding to create a separate tiling over each potential weight feature to create the CMDP basis space.

We used the same 15 tasks used in the code release of Svetlik et al. [10] to form the action space $\mathcal{A}^C$. The set of terminal states S_f^C were all states where the learning agent could achieve a return of at least 2000 on the target task. Figure 2(a) shows CMDP learning curves for Ms. Pac-Man using value function transfer and Figure 2(b) shows the curves using reward shaping. The results again clearly show that curriculum policies can be learned, and that such policies are more useful than training directly on the target task. In addition, we compared the reward shaping approach with that of Svetlik et al.[10], and found that a much better curriculum is possible in this more complex domain.[1]

Finally, we also study the effect of the hyperparameter that controls when to finish training on a source task. For the previous two experiments in Ms. Pac-Man, training on a source was stopped after 35% of the max possible return in the task was achieved, to replicate the experimental conditions of [10]. Since their approach precomputes a curriculum and does not model the state of the learning agent's progress, this termination condition must be carefully chosen to ensure something can be learned in each source task. In contrast, with our approach, we can train on source tasks for an arbitrarily small amount of time, as the curriculum policy can learn to reselect a task if additional experience in that task is required.

[1]Our results are based on a reproduction of their experiments using their publicly released code. Interestingly, we get slightly better results for their method than they report in their paper.

In Figure 2(c), we reproduce the continuous state representation CMDP learning curves using value function transfer from Figure 2(a) and reward shaping from Figure 2(b). These are denoted in the figure by "(return-based)", and train on sources until 35% of the max return is achieved. We compare them against an approach that is identical to "(return-based)" approaches, but that trains for 5 episodes on a task at a time. These CMDP learning curves are denoted with "(small fixed)." The results show that agents do not need to train for a long time or to convergence on source tasks, and that our approach can adapt to this hyperparameter setting.

4 Conclusion

In this paper, we showed that a more general representation of a curriculum than previous work, a *curriculum policy*, can be learned. The key challenge of learning a curriculum policy is creating a CMDP state representation that allows efficient learning. We extended the original curriculum MDP definition to handle multiple types of transfer learning algorithms, and described how to construct CMDP representations for both discrete and continuous domains to faciliate such learning. Finally, we demonstrated that curriculum policies can be learned on a gridworld and pacman domain. The results show that our approach is successful at creating curricula that can train agents to perform on a target task as fast or faster than existing methods. Furthermore, our approach is robust to multiple learning agent types, multiple transfer learning algorithms, and different CMDP representations.

One limitation of our approach is that learning a full curriculum policy can take significantly more experience data than learning the target policy from scratch. An important direction for future work is investigating the extent to which this cost can be amortized by reusing learned curricula for multiple, similar target tasks. The contributions of this paper are an essential prerequisite for such an investigation. Another interesting direction for future work is to examine the extent to which the methods presented here generalize to policy-gradient-based approaches and transfer learning algorithms, in addition to the value-function-based algorithms that were used in all of our experiments.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM, 2009.

[2] Felipe Leno Da Silva and Anna Helena Reali Costa. Object-oriented curriculum generation for reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

[3] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1515–1528, Stockholmsmssan, Stockholm Sweden, July 2018.

[4] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

[5] A. Lazaric. Transfer in reinforcement learning: a framework and a survey. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State of the Art*. Springer, 2011.

[6] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source task creation for curriculum learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, May 2016.

[7] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2017.

[8] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[9] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[10] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. Automatic curriculum graph generation for reinforcement learning agents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, February 2017.

[11] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

Habits as a Function of Choice Frequency: A Novel Experimental Approach to Study Human Habits

Stephan Nebe
Zurich Center for Neuroeconomics,
Department of Economics,
University of Zurich
Zürich, Switzerland
stephan.nebe@econ.uzh.ch

André Kretzschmar
Hector Research Institute of Educational
Sciences and Psychology,
University of Tübingen
Tübingen, Germany
andre.kretzschmar@uni-tuebingen.de

Philippe N. Tobler
Zurich Center for Neuroeconomics,
Department of Economics,
University of Zurich
Zürich, Switzerland
phil.tobler@econ.uzh.ch

Abstract

In habitual behavior, context information elicits responses without active deliberation. Habits reduce the cognitive load in everyday life and abound in maladaptive conditions such as drug addiction. Due to the ubiquity and clinical importance of habits, it is essential to study them in the lab. However, recent research revealed that the current experimental approaches to human habitual behavior lack validity, replicability and consistency. Previous experimental arrays examining habitual control often overlooked that habits by definition should be independent from value, that is, the consequences of an action should neither be considered nor even represented when performing a habit. Instead, habit strength should be proportional to the frequency of performing a given behavior without reinforcement. Yet, it remained unclear whether such a framework can be studied experimentally.

For this ongoing study, we have designed a new experimental task, which realigns the empirical approach to habits with the theoretical, value-free, foundations. Our task assesses habitual control as a function of previous choice frequency in addition to and even in the complete absence of reinforcement. In a pilot study, we tested the influence of previous choice frequency on preferences in binary decisions. Surprisingly, previous choice frequency affected choices in the opposite direction of the assumed habit strength in a learning task with reinforcement or not at all in the focal task without reinforcement. These results highlight the difficulties of assessing human habits experimentally and of aligning practice with theory of habits.

Keywords: habit, behavioral control, value-based decision making, experimental psychology

Acknowledgements

We thank Ifat Levy, Marc Guitart-Masip, Sam Gershman, Tal Yarkoni, and Lydia Hellrung for fruitful discussions regarding the design of the experimental paradigm. This work was supported by grants 100014_165884 and 100019_176016 from the Swiss National Science Foundation.

1 Habitual behavior

We typically think of our behavior as goal-directed and purposeful. However, research suggests that a large part of our everyday behavior is habitual rather than goal-directed [1]. Habits have been defined as counterpart of goal-directed behavior [2]. Per definition, goal-directed behavior is guided by knowledge about the contingency between an action and its outcome and by knowledge about the incentive value of this outcome [3]. Thereby, the goal-directed system enables flexible behavior based on beliefs about environmental contingencies and the forward-looking achievement of valuable goals. In contrast, habitual behavior corresponds to stimulus-response associations, which evolve over time and in which stimuli eventually trigger behavior inflexibly. In theory, behavior is goal-directed when executed for the first time or in new contexts. With repetition of actions in rather stable environments, behavioral control can get detached from the knowledge of contingencies and outcomes values [4]. Thus, context-dependent automaticity ensues. This process of habituation requires many repetitions, making habits slow to learn and slow to forget.

Yet, habits are ubiquitous in our daily life, from our morning routines in the bathroom to the leisure activities with which we spend our evenings. Previous research has associated habitual control with a wide range of behaviors, for example frequency of physical exercise [5]. Furthermore, there is cumulating evidence for habits to play a role in pathologically altered choice behavior in substance use disorders [6], [7]. However, habits are not disadvantageous in general. They free cognitive resources for other tasks, because habitual control involves merely executing previously repeated behavior in a computationally simple and effortless manner.

In view of the presumed ubiquity of habits, it has proven surprisingly difficult to study them in the lab, at least with humans [8]. Classical examinations of goal-directed and habitual control investigated either of the two defining criteria of goal-directed behavior, thus using outcome devaluation or contingency degradation tasks, respectively. These tasks study how a learned instrumental response changes when the contingency between action and outcome or the value of the outcome change. If response rates decrease when the outcome is no longer valued or the actions are no longer necessary to achieve the outcome, actions are taken to be goal-directed. In contrast, if the originally learned response perseverates, behavior is taken to be habitual [9].

Overtraining of an instrumental response in stable conditions or training in strong motivational or frustrating states (e.g. hunger during training) is thought to lead to a shift in behavioral control from goal-directed to habitual. Animal research has shown this shift time and again over the last decades reaching back to Tolman's studies of cognitive maps [10]. In contrast, only one study was able to demonstrate habituation of behavior via overtraining in a human laboratory experiment [11]. Unfortunately, this finding could not be replicated in two studies with a similar setup and larger sample sizes [8] questioning the original findings' validity.

Researchers recently tried to overcome previous shortcomings by using a different experimental approach involving sequential Markov decision tasks. These tasks operationalize habitual and goal-directed behavior as model-free and model-based reinforcement learning, respectively [12]. The computational modeling approach accompanying these paradigms makes the relative contribution of model-free and model-based reinforcement learning measurable.

In contrast to habits, model-free reinforcement learning is driven entirely by the incentive value of outcomes associated with behavior. In-keeping with this contrast, previous studies investigating the overlap of model-free reinforcement learning and devaluation insensitivity found only statistically non-significant small to very small associations, thereby contradicting the notion that they should both operationalize habitual behavior [13]–[15]. These findings indicate that model-free reinforcement learning is neither conceptually nor empirically equivalent to habitual behavior and reinforce the need for a better experimental approach to the study of habits.

2 Experimental paradigm

We have designed a new experimental paradigm manipulating habit strength. It has been argued that

habitual control might be driven solely by the frequency of choice irrespective of any outcomes associated with behavior [16]. Indeed, many everyday choices do not yield their outcomes immediately. Therefore, the new experimental paradigm comprised two binary-choice tasks. While the first task was a reinforcement-learning paradigm manipulating choice frequency independently from rewards (see Figure 1), the second task examined the possibility of developing habits without any obvious reinforcing outcomes of choice behavior.

The first task began with a training phase, in which participants deduced optimal choice behavior via trial-and-error from the reward feedback they got (winning 1-5 points). Two stimuli each yielded two, three, or four points. One stimulus per reward level was paired 20 times with a stimulus worth less and 10 times with a stimulus worth more points and vice versa for the other stimulus. Thus, participants saw each stimulus equally often, but chose one stimulus more often during training than the other one of the same reward level. Full feedback about the chosen as well as unchosen option ensured that participants were similarly certain about the reward levels associated with rarely and more frequently chosen stimuli. In a subsequent free-choice phase in extinction, test trials combined for the first time the two stimuli which during training were associated with the same reward level but different choice frequencies. If higher choice frequency translates into stronger habits, participants should prefer the stimulus chosen more often in training to the other stimulus of the same reward level.

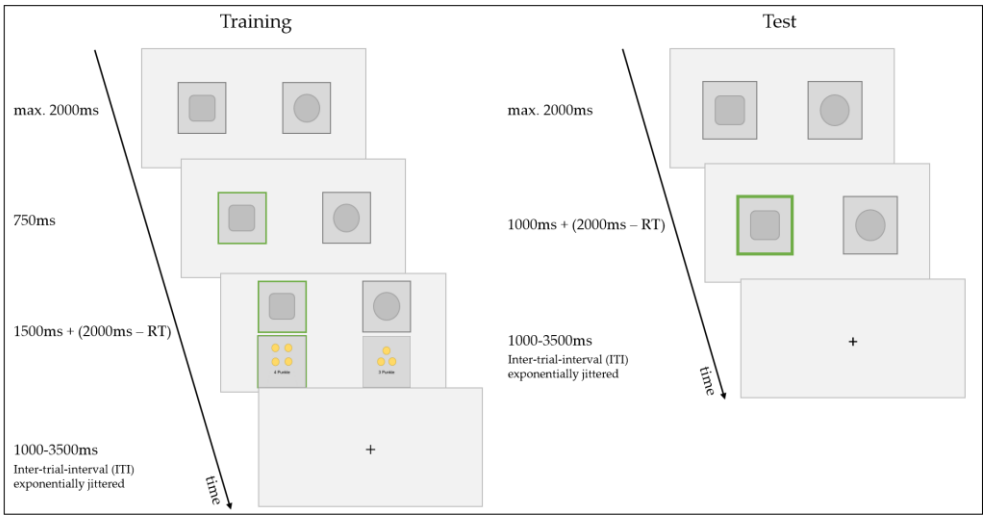


Figure 1. Trial sequence of the training and test phase of the first task (with outcomes).

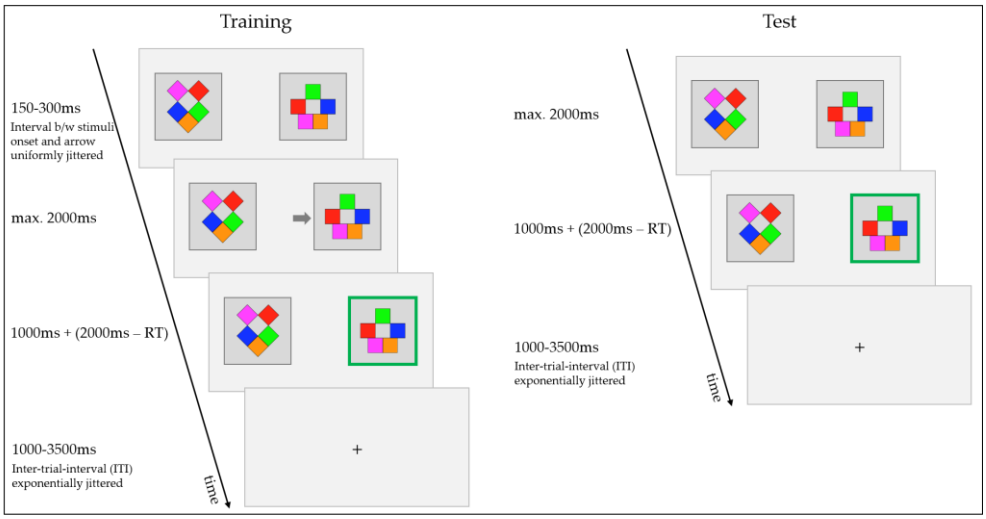


Figure 2. Trial sequence of the training and test phase of the second task (without outcomes).

The second task examined the possibility of developing habits from behavior that has never been paired with reinforcement. It began with an instructed-choice phase, in which participants viewed two abstract stimuli and were instructed to choose the one highlighted by an arrow as quickly and accurately as possible. Unbeknownst to them, the frequencies of highlighting the different stimuli were not uniformly distributed. In one pair of stimuli, the arrow highlighted both stimuli equally often, whereas in the other pair of stimuli one stimulus was highlighted in 80% of trials. The two stimulus pairs were presented in ten alternating blocks of 12 trials each. The presentation locations of the stimuli were fixed within participants but varied between participants. Critically, there were no outcomes associated with the various stimuli. In the test phase, participants were instructed to choose as quickly as possible one of the presented stimuli. Without any criterion to guide choice

during the free-choice test phase, random decisions or choice according to simple rules (e.g. always choosing the left stimulus) could be expected. In contrast, if habituation of behavior is indeed possible in this value-free context, participants are expected to choose the stimulus more often that has been highlighted and therefore selected more frequently during the instructed-choice phase.

3 Results of piloting

Twenty participants took part in a pilot study. In the training phase of the task with reinforcement, participants chose some stimuli more often than others because the different stimuli were associated with different reward levels. In the test phase, each stimulus was paired with every other. Participants showed a strong tendency towards higher valued stimuli (Figure 3, left). Accordingly, logistic mixed effects regression of the behavioral data revealed a strong effect of reward level (z=21.17, p<.001). More importantly, we also found an effect of previous choice-frequency (z=8.82, p<.001) in the test phase. Surprisingly, the effect of choice frequency was in the opposite direction of the hypothesis: When facing two stimuli of the same reward level, participants chose the previously less frequently selected stimuli more often (Figure 3, right). Possibly, the policy of choosing these stimuli has a higher novelty for participants and, thus, is preferred, but we need to interpret this finding with caution due to the small sample size of this pilot study.

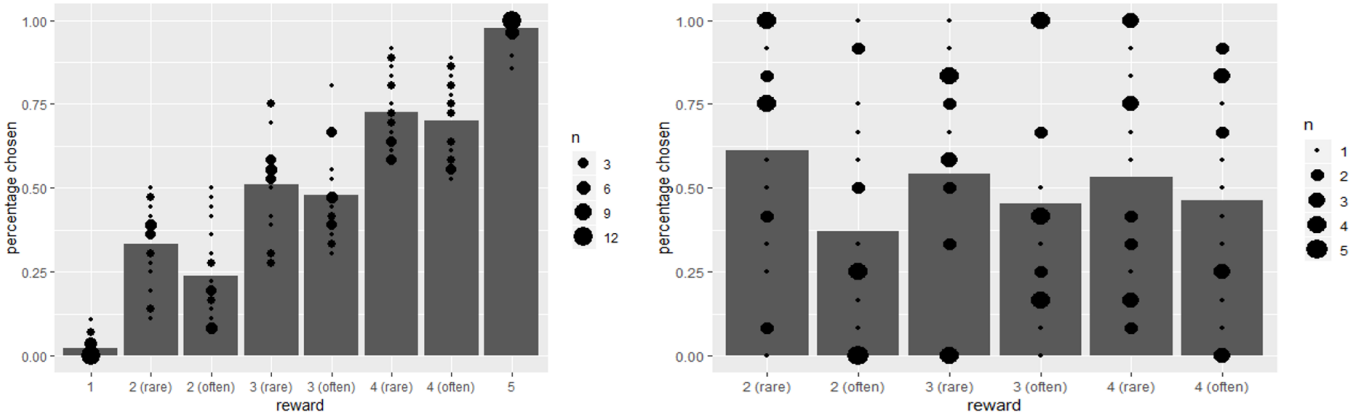


Figure 3. Participants' preferences during the test phase of the first task depending on the associated reward level (1-5 points) and the choice frequency during training (often or rarely chosen) in all trials (left) and in those trials only, which compared stimuli of the same reward level.

In the training of the task without reinforcement, there were two pairs of stimuli with choice probabilities of 50/50% and 80/20%, respectively. In a subsequent free-choice phase, participants were instructed to choose intuitively between the presented stimuli. Again, each stimulus of this task was presented with every other. A logistic mixed effects regression did not reveal an effect of prior choice frequency on decision making during the test phase (z=1.00, p=.316; see Figure 4). During training, stimulus presentation locations were fixed for each participant, theoretically enabling automatization of choice behavior in the 80/20% stimulus

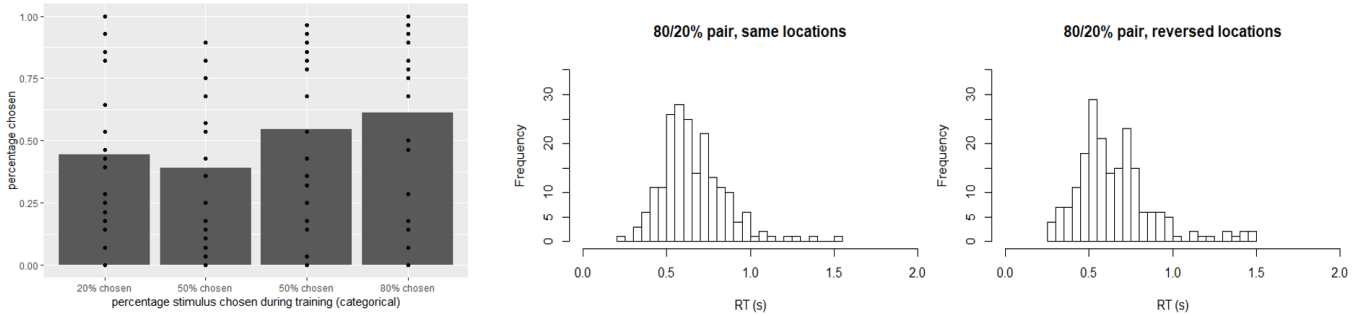


Figure 4. Participants' choice behavior during the test phase of the second task depending on the choice frequency during training (left). Response times in trials showing the 80/20% stimulus pair in the same (middle) or reversed (right) locations did not differ from each other.

pair. In contrast, stimulus presentation location were balanced within subject during the test phase. Therefore, half of the trials including both stimuli of the previous 80/20% pair were exact copies of the respective training trials according to stimulus presentation, whereas the other half showed stimuli in exact opposition compared to training. If there would be habituation in this context, the reversed stimulus presentation location should have evoked switch costs in terms of response times (which arguably are more sensitive to habitual behavior than overt choice; [17]). Response times did not differ between originally trained and reverse location (pseudo-median of location shift between distributions=-0.02, 95% CI=[-0.05,0.01], p=.195; see Figure 4). Thus, habits were not evident in this task without reinforcement reiterating the possibility that habitual behavior can only ensue from actions that have been goal-directed in the beginning.

4 Discussion and outlook

These results indicate a process other than habituation leading participants' choice behavior. Possibly, the rarely chosen stimuli have a higher novelty for the participants, thus guiding choice towards themselves in the first task. In contrast, habitual behavior could not be induced in an experimental setting without reinforcement. Possibly, reinforcement is needed to boost learning of habits. The experimental setup of the tasks will be adapted regarding the amount of training and the response time window to test whether habituation of behavior will ensue under conditions, which are known to favor the development of habits.

References

[1] B. Verplanken and S. Orbell, "Reflections on Past Behavior: A Self-Report Index of Habit Strength," *Journal of Applied Social Psychology*, vol. 33, no. 6, pp. 1313–1330, Jul. 2006.
[2] R. J. Dolan and P. Dayan, "Goals and Habits in the Brain," *Neuron*, vol. 80, no. 2, pp. 312–325, Oct. 2013.
[3] A. Dickinson and B. Balleine, "Motivational control of goal-directed action," *Animal Learning & Behavior*, vol. 22, no. 1, pp. 1–18, 1994.
[4] B. Verplanken, H. Aarts, A. Knippenberg, and C. Knippenberg, "Attitude Versus General Habit: Antecedents of Travel Mode Choice," *Journal of Applied Social Psychology*, vol. 24, no. 4, pp. 285–300, Feb. 1994.
[5] L. Fleig *et al.*, "From intentions via planning and behavior to physical exercise habits," *Psychology of Sport and Exercise*, vol. 14, no. 5, pp. 632–639, Sep. 2013.
[6] K. D. Ersche *et al.*, "Carrots and sticks fail to change behavior in cocaine addiction," *Science*, vol. 352, no. 6292, pp. 1468–1471, Jun. 2016.
[7] V. Voon *et al.*, "Disorders of compulsivity: a common bias towards learning habits," *Molecular Psychiatry*, vol. 20, no. 3, pp. 345–352, Mar. 2015.
[8] S. de Wit *et al.*, "Shifting the balance between goals and habits: Five failures in experimental habit induction," *Journal of Experimental Psychology: General*, vol. 147, no. 7, pp. 1043–1065, 2018.
[9] C. D. Adams and A. Dickinson, "Instrumental responding following reinforcer devaluation," *The Quarterly Journal of Experimental Psychology Section B*, vol. 33, no. 2, pp. 109–121, May 1981.
[10] E. C. Tolman, "Cognitive maps in rats and men," *The Psychological Review*, vol. 55, no. 4, pp. 189–208, 1948.
[11] E. Tricomi, B. W. Balleine, and J. P. O'Doherty, "A specific role for posterior dorsolateral striatum in human habit learning," *European Journal of Neuroscience*, vol. 29, no. 11, pp. 2225–2232, Jun. 2009.
[12] N. D. Daw, S. J. Gershman, B. Seymour, P. Dayan, and R. J. Dolan, "Model-Based Influences on Humans' Choices and Striatal Prediction Errors," *Neuron*, vol. 69, no. 6, pp. 1204–1215, Mar. 2011.
[13] E. Friedel, S. P. Koch, J. Wendt, A. Heinz, L. Deserno, and F. Schlagenhauf, "Devaluation and sequential decisions: linking goal-directed and model-based behavior," *Frontiers in Human Neuroscience*, vol. 8, Aug. 2014.
[14] C. M. Gillan, A. R. Otto, E. A. Phelps, and N. D. Daw, "Model-based learning protects against forming habits," *Cognitive, Affective, & Behavioral Neuroscience*, vol. 15, no. 3, pp. 523–536, Sep. 2015.
[15] Z. Sjoerds *et al.*, "Slips of Action and Sequential Decisions: A Cross-Validation Study of Tasks Assessing Habitual and Goal-Directed Action Control," *Front. Behav. Neurosci.*, vol. 10, 2016.
[16] K. J. Miller, A. Shenhav, and E. A. Ludvig, "Habits without values," *Psychological Review*, 2019.
[17] D. Luque, S. Molinero, P. Watson, F. J. López, and M. Le Pelley, "Measuring habit formation through goal-directed response switching," *PsyArXiv*, 2019.

Pseudo-Learning Rate Modulation by the Forgetting of Action Value when Environmental Volatility Changes

Susumu Oshima
Department of Cognitive and Psychological Sciences
Graduate School of Informatics, Nagoya University
Aichi 464-8601, Japan
s.oshima@prsnl.onmicrosoft.com

Kentaro Katahira
Department of Cognitive and Psychological Sciences
Graduate School of Informatics, Nagoya University
Aichi 464-8601, Japan
katahira.kentaro@b.mbox.nagoya-u.ac.jp

Abstract

Many studies have reported that animals modulate their speed of learning, measured by estimated learning rate, to cope with the differing degree of stability in reward structure. While these studies have assumed some neural computation of direct modulation, the actual process underlying it is not clearly understood. The present study proposes the possibility that the observed difference in estimated learning rates may not be a consequence of the learning rate modulation, but could be statistical artifacts of other characteristics of learning, such as forgetting of the learned value of choices. The simulated probabilistic reversal learning tasks used in those studies revealed that the apparent learning rate modulation emerges when the learner has been forgetting action value, and yet this was not considered in parameter estimation. The same effect arises, albeit to a lesser degree, when the learning rate is asymmetric by prediction error, as well as when the learner has a tendency to perseverate past choices. The findings call for re-evaluation of past studies, and pose a question about the common practice of fitting only models with task-relevant components to behavior.

Keywords: forgetting process, learning rate, parameter estimation, statistical artifacts, volatility

Acknowledgements

This study was supported by Grant-in-Aid for Scientific Research (B) (18KT0021 to KK).

1 Introduction

Understanding how animals learn in uncertain environments is one of the goals of learning psychology. This uncertainty can be divided into two forms: *expected uncertainty*, caused by noise in an action–outcome contingency (that is, reward is given stochastically), and *unexpected uncertainty*, caused by volatility of the contingency (that is, reward probability changes over time) (Bland & Schaefer, 2012; Yu & Dayan, 2005). These two forms of uncertainty differentiate optimal choice strategy: averaging over a long history of outcomes for expected uncertainty and focusing on the most recent outcomes for unexpected uncertainty. In terms of reinforcement learning, an optimal learner must implement a higher *learning rate* when the contingency is volatile than when the only source of uncertainty is noise (Courville, Daw, & Touretzky, 2006; Preuschoff & Bossaerts, 2007). Behavioral studies have reported that the estimated learning rate was actually higher when the reward environment was volatile than when it was stable (Behrens, Woolrich, Walton, & Rushworth, 2007; McGuire, Nassar, Gold, & Kable, 2014; Massi, Donahue, & Lee, 2018). Furthermore, the amount of difference between the two learning rates was correlated with trait anxiety (Browning, Behrens, Jocham, O'Reilly, & Bishop, 2015).

These studies assumed that an animal's brain implements a mechanism to modulate the learning rate, and many of them tried to identify responsible regions with brain imaging. However, it is not clear whether the brain directly modulates the learning rate or the observed difference was a by-product of other mechanisms that indirectly affected the estimated learning rate. Here we propose the possibility that the learning rate modulation may be, at least partly, caused by the forgetting of learned action value (or Q-value), represented as

$$Q_{t+1}(a_j) = (1-\alpha_F)Q_t(a_j) \quad \text{for all } j \neq i,$$

when action $a_i (i \neq j)$ was chosen at trial t. Here $Q_t(a_j)$ is the action value of a_j at t and α_F is a forgetting rate parameter. This forgetting Q-learning model was proposed on a psychological assumption (Roth & Erev, 1995), and proved to fit the behavior of animals better than the standard Q-learning model (Ito & Doya, 2009; Katahira, Yuki, & Okanoya, 2017; Toyama, Katahira, & Ohira, 2017; Worthy, Hawthorne, & Otto, 2013). As far as we are aware, no study has investigated the effect of forgetting on learning rate estimation, or incorporated it into learning rate estimation. We also briefly show the effect by two other variants of the Q-learning model. The first, Q-learning with asymmetric learning rates, is represented as

$$\begin{cases} Q_{t+1}(a_i) = Q_t(a_i) + \alpha^+\big(R_t - Q_t(a_i)\big) & \big(R_t - Q_t(a_i) \geq 0\big), \\ Q_{t+1}(a_i) = Q_t(a_i) + \alpha^-\big(R_t - Q_t(a_i)\big) & \big(R_t - Q_t(a_i) < 0\big), \end{cases}$$

where R_t represents the reward at t, α^+ represents the learning rate when prediction error $R_t - Q_t(a_i)$ is positive, and α^- represents the learning rate when it is negative (Lefebvre, Lebreton, Meyniel, Bourgeois-Gironde, & Palminteri, 2017; Niv, Edlund, Dayan, & O'Doherty, 2012). The second variant, Q-learning with perseveration, is represented as follows when coupled with the widely used softmax action selector (Worthy, Pang, & Byrne, 2013):

$$C_{t+1}(a_i) = C_t(a_i) + \tau\big(I(a(t)=a_i) - C_t(a_i)\big), \quad \text{where } I(a(t)=a_i) = \begin{cases} 1 & (a_i \text{ was chosen at } t), \\ 0 & (\text{otherwise}), \end{cases}$$

$$\Pr(a_1) = \frac{1}{1+\exp\big\{-\beta\big(Q_t(a_1)-Q_t(a_2)\big) - \varphi\big(C_t(a_1)-C_t(a_2)\big)\big\}}, \quad \Pr(a_2) = 1 - \Pr(a_1).$$

Here, C, τ, and φ represent choice kernel, choice learning rate, and degree of perseveration, respectively.

2 Method

We simulated probabilistic reversal learning tasks similar to the one in Browning et al. (2015), which consists of two blocks: stable and volatile blocks. Each block contained 250 trials, and in each trial a simulated agent chose one of two choices $a_i (i = 1, 2)$ to get reward R_t, which was either 1 or 0 (no reward). In the stable block, a_1 and a_2 gave reward with 75% chance and 25% chance, respectively. In the volatile block, at first a_2 gave reward with 80% chance and a_1 with 20%; however, the probabilities switched on every 20 trials except for the final 10 trials. The simulation ran the forgetting Q-learning model coupled with the softmax action selector in the task, and this was repeated 200 times for each forgetting rate. α_F ranged from 0 to 1.0, in increments of 0.01 for both stable-first-volatile-last and volatile-first-stable-last order. The learning rate in each block was separately estimated by fitting the data to the standard Q-learning model, as performed in many studies. In this process, fixed-effect maximum likelihood estimation was applied to the data produced by each α_F and each block order. The whole process was also applied to other two models—the asymmetric learning rate and perseveration models—by adjusting the process of parameter manipulation. Values of $Q_1 = 0, \alpha = 0.2$, and $\beta = 5.0$ were used except for the asymmetric learning rate model, in which learning rates were varied from $\alpha^+ = \alpha^- = 0.20$ to $\alpha^+ = 0.01, \alpha^- = 0.39$ by decrementing α^+ and incrementing α^- by 0.002 simultaneously.

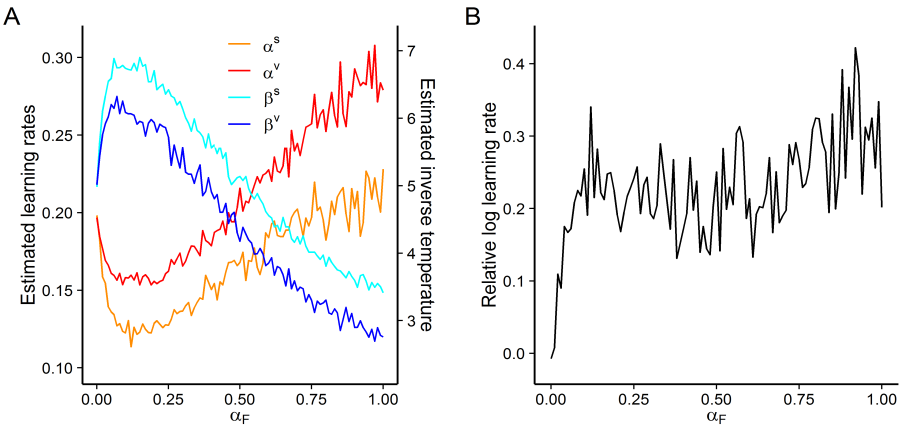


Figure 1: (A) Estimated parameters of forgetting Q-learning for each block. $\alpha^s, \alpha^v, \beta^s$, and β^v correspond to the learning rate in the stable block, that in the volatile block, the inverse temperature in the stable block, and that in the volatile block, respectively. (B) Log-transformed difference of the estimated learning rates between the two blocks (log (learning rate in volatile block) − log (learning rate in stable block)), as a function of α_F in the forgetting Q-learning.

3 Results

Figure 1 (A) shows how the degree of forgetting affected estimated parameters. Although the real learning rate (that is, that used to produce the data) was constant, the estimated learning rate was consistently higher in the volatile block, while the inverse temperature exhibited the opposite pattern. Figure 1 (B) shows how the difference of the learning rate changed when α_F increased, represented by the relative log scale used in Browning et al. (2015) to show correlation with trait anxiety. In this scale, 0 means that the learning rates of the two blocks were equivalent, while 1.0 means that the learning rate in the volatile block was 2.718 times higher than that in the stable block. Although it did not reach 1.0, which is the value the least anxious subjects in Browning et al. (2015) exhibited, the difference became larger as α_F increased. Figure 2 is comparable to Figure 1 (B) but it was produced by the asymmetric learning rate (Figure 2 (A)) and perseveration (Figure 2 (B)) model. Although the differences were smaller than those of the forgetting model, they exhibited the same pattern: they became larger as the normalized learning rate asymmetry or τ increased, in both models.

4 Discussion and Conclusion

We demonstrated that the forgetting of action value can cause apparent learning rate modulation when the environmental volatility changes, and, to a lesser degree, asymmetry in learning rates and perseveration can do the same. This

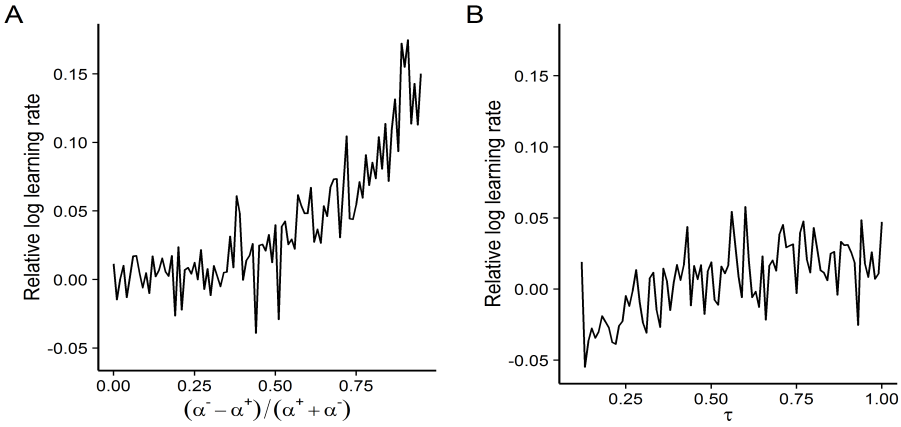


Figure 2: (A) The log-transformed difference of the estimated learning rates as a function of normalized learning rate asymmetry $(\alpha^- - \alpha^+)/(\alpha^+ + \alpha^-)$ in the asymmetric learning rate model. (B) The log-transformed difference of the estimated learning rates as a function of τ in the perseveration model.

suggests that the results of studies that reported learning rate modulation by fitting a standard Q-learning model may be statistical artifacts caused by neglecting the forgetting, and possibly other factors such as asymmetric learning rate or perseveration, though this cannot be confirmed.

A legitimate question is why the pseudo-modulation occurred. The key to answering the question is, in addition to the contingency switch, the initial conditions of the action values (Q-values). In stable-first-volatile-last order, the stable block starts with $Q_1 = 0$ for both choices; however, before entering into the volatile block, the Q-values approach the expected values of each choice. In contrast, in volatile-first-stable-last order, the volatile block starts with $Q_1 = 0$; however, as the volatile block repeatedly switches the contingency, the effect of these first trials on parameter estimation will be offset by the subsequent switches, which lead to different Q-values. However, they will not converge to the expected value of each choice before entering into the stable block, particularly because the last part of the volatile block only contained 10 trials (this was also true in Browning et al. (2015)). Importantly, if the initial Q-values are equally low the forgetting mostly slows the learning down by reducing the learned value of the better choice. (Note that either choice can be chosen with enough chance in the early stages of learning in this case.) Compared to this, reducing the value of the worse choice is not important because the expected Q-value for this choice will be low even without forgetting, and thus reducing it will not contribute much to the action selection. In contrast, when the initial Q-value of the choice that was formerly preferred is higher, the forgetting starts to reduce the Q-value once the current better choice is chosen, leading to quicker learning. However, forgetting also causes the Q-value of the formerly worse choice to regress to 0, provided the chance of making this choice has been low enough in the stable block. This is disadvantageous for quicker learning, because it leads to a larger difference between initial Q-values and thus causes the new better choice not to be chosen. However, from the perspective of standard Q-learning (recall that the finding emerged when the data produced by the forgetting model were fitted to standard Q-learning), this larger difference can be interpreted as a higher inverse temperature β, which partly nullifies the disadvantage. Once the learner switches its preferred choice, the forgetting expands the difference between Q-values. This can appear to be a sign of a higher learning rate, but also will be interpreted as a sign of a higher β.

We also analytically let standard Q-learning minimize the difference between the expected values of the choice probabilities for the first three choices in each block with the forgetting Q-learning, as follows:

Proposition. Let P_H and P_t^s be the reward probability for better choice (a_1 for the stable block and a_2 for the volatile block) and choice probability of a_1 at trial $t(t = 1, 2, 3)$ in the stable block produced by the softmax action selector. Also let $\alpha_0, \beta_0, \alpha^s$, and β^s be fixed learning rate of the forgetting Q-learning model, fixed inverse temperature of the forgetting Q-learning, learning rate of the standard Q-learning in the stable block that minimizes the difference in expected choice prediction with the forgetting Q-learning, and inverse temperature of the standard Q-learning in the stable block that minimizes that, respectively. In the same manner, P_t^v, α^v, and β^v are defined for the volatile block, and the reward probability for the worse choice is $1 - P_H$. Assume that $Q_1 = 0$ for both models in the stable block, while in the volatile block (trial number restarts from 1), $E(Q_1(a_1)) = P_H$ and $E(Q_1(a_2)) = 0$ for the forgetting Q-learning and $E(Q_1(a_1)) = P_H$ and $E(Q_1(a_2)) = 1 - P_H$ for the standard Q-learning, where $E(\cdot)$ means the expected value. In addition, assume that $\alpha_F = \alpha_0$ as well as conditions **P**, **S**, **V1**, and **V2**.
P: $E\big(Q_t(a_1) - Q_t(a_2)\big) = E\big(Q_t^{'}(a_1) - Q_t^{'}(a_2)\big) \Rightarrow P_t = P_t^{'}$: This is required because, as a result of the nonlinearity of the softmax function, the same values of $E(Q_t(a_1) - Q_t(a_2))$ do not necessarily mean that the choice probabilities are the same.
S: $P_2^s < P_H$: This is satisfied unless α^s and β^s are too high, and ensures that the choice predictions of the two models differ in the stable block.
V1: $\beta^v = \beta_0 P_H/(2P_H - 1)$: This ensures that the P_1^v for both models are the same.
V2: $P_1^v = P_H$: This ensures that the P_2^v for both models are the same. **V1** and **V2** also ensure that the forgetting Q-learning will explore lower-valued actions, which is important for the forgetting to work in the limited trials of three.

Then the following holds:

$$\alpha^s < \alpha_0 < \alpha^v,\ \beta^s = \beta_0 < \beta^v$$

Sketch. $E(Q_3(a_1) - Q_3(a_2))$ for the forgetting and standard Q-learning (denoted as $E(\Delta Q_3^F)$ and $E(\Delta Q_3)$) in the stable block are as follows:

$$\begin{cases} E(\Delta Q_3^F) = 0.5\alpha_0(4P_H - 2\alpha_0 P_H + 2P_2 + \alpha_0 - 3) \\ E(\Delta Q_3) = 0.5\alpha^s(4P_H - \alpha^s P_H + 2P_2 - \alpha^s P_2 + \alpha^s - 3) \end{cases}$$

By condition **S**, the difference between the choice predictions can only be minimized by either decreasing α^s or β^s; however, both operations affect the currently calculated Q-values. Numerical analyses showed that the effect of the former operation is smaller. If the former operation is applied and the difference in P_2^s is ignored[1], α^s can be represented

[1]In the setting used in Section 2, except for $\beta_0 \approx 1.465$ to meet condition **V2**, ε^s is around 0.01 and the differences of P_2^s and P_3^s are about 9×10^{-4} and 1.7×10^{-3} in probability. While the difference in P_2^s can be completely nullified by expanding β^s to $\beta_0\alpha_0/(\alpha_0 - \varepsilon^s)$, this in turn causes the differences in $\Delta Q_3^F - \Delta Q_3$ to not be reducible to less than $0.5\alpha_0(P_H - P_2^s)$, which is not negligible.

as $\alpha_0 - \varepsilon^s$ with

$$\varepsilon^s = \frac{2\alpha_0 A - B + \sqrt{4\alpha_0^2 A(2P_H - 1) - 4\alpha_0 AB + B^2}}{2A} > 0,$$
$$\text{where } A := P_H + P_2^s - 1, B := 4P_H + 2P_2^s - 3.$$

In a similar way, α^v can be represented as $\alpha_0 - \varepsilon^v$ with

$$\varepsilon^v = \frac{-P_H(1 - \alpha_0 C) + \sqrt{P_H^2(1 - \alpha_0 C)^2 - P_H CD}}{C} < 0,$$
$$\text{where } C := 2P_2^v P_H - P_2^v - P_H + 1,$$
$$D := P_2^v \left\{\alpha_0^2 P_H(P_H - 1) + \alpha_0(P_H^2 + P_H - 1) - P_H\right\} + \alpha_0 P_H(1 - P_H).$$

□

The findings not only call for further research to elucidate the mechanism underlying the learning rate modulation, but also pose a more general question about the common practice of fitting behavioral data to standard Q-learning and/or models with directly task-relevant components only. While computational modeling helps to uncover otherwise hidden processes of neural computation, problems arise if incomplete models are fitted (Katahira, 2018; Nassar & Gold, 2013). Fitting models that incorporate task-irrelevant characteristics not only minimizes the risk of statistical artifact, but will also help to increase the understanding of the actual computation of the brain. However, the degree of difference in the estimated learning rates did not reach that of the least anxious subjects in Browning et al. (2015), and as shown in Figure 1 (A), forgetting also has effects on the inverse temperature, which may suggest that forgetting alone is not enough to explain the learning rate modulation. Further research is required to elucidate the actual process of learning rate modulation.

References

Behrens, T., Woolrich, M., Walton, M., & Rushworth, M. (2007). Learning the value of information in an uncertain world. *Nature Neuroscience*, **10**(9), 1214-1221.

Bland, A. R., & Schaefer, A. (2012). Different varieties of uncertainty in human decision-making. *Frontiers in Neuroscience*, **6**(85).

Browning, M., Behrens, T., Jocham, G., O'Reilly, J., & Bishop, S. (2015). Anxious individuals have difficulty learning the causal statistics of aversive environments. *Nature Neuroscience*, **18**(4), 590-596.

Courville, A., Daw, N., & Touretzky, D. (2006). Bayesian theories of conditioning in a changing world. *Trends in cognitive sciences*, **10**(7), 294-300.

Ito, M., & Doya, K. (2009). Validation of decision-making models and analysis of decision variables in the rat basal ganglia. *Journal of Neuroscience*, **29**(31), 9861-9874.

Katahira, K. (2018). The statistical structures of reinforcement learning with asymmetric value updates. *Journal of Mathematical Psychology*, **87**, 31-45.

Katahira, K., Yuki, S., & Okanoya, K. (2017). Model-based estimation of subjective values using choice tasks with probabilistic feedback. *Journal of Mathematical Psychology*, **79**, 29-43.

Lefebvre, G., Lebreton, M., Meyniel, F., Bourgeois-Gironde, S., & Palminteri, S. (2017). Behavioural and neural characterization of optimistic reinforcement learning. *Nature Human Behaviour*, **1**(4), 0067.

Massi, B., Donahue, C., & Lee, D. (2018). Volatility facilitates value updating in the prefrontal cortex. *Neuron*, **99**(3), 598-608.e4.

McGuire, J., Nassar, M., Gold, J., & Kable, J. (2014). Functionally dissociable influences on learning rate in a dynamic environment. *Neuron*, **84**(4), 870-881.

Nassar, M., & Gold, J. (2013). A healthy fear of the unknown: perspectives on the interpretation of parameter fits from computational models in neuroscience. *Journal of Neuroscience*, **37**(29), 7023-7035.

Niv, Y., Edlund, J., Dayan, P., & O'Doherty, J. (2012). Neural prediction errors reveal a risk-sensitive reinforcement-learning process in the human brain. *Journal of Neuroscience*, **32**(2), 551-562.

Preuschoff, K., & Bossaerts, P. (2007). Adding prediction risk to the theory of reward learning. *Annals of the New York Academy of Sciences*, **1104**, 135-146.

Roth, A., & Erev, I. (1995). Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior*, **8**(1), 164-212.

Toyama, A., Katahira, K., & Ohira, H. (2017). A simple computational algorithm of model-based choice preference. *Cognitive, Affective, & Behavioral Neuroscience*, **17**(4), 764-783.

Worthy, D., Hawthorne, M., & Otto, A. (2013). Heterogeneity of strategy use in the iowa gambling task: a comparison of win-stay/lose-shift and reinforcement learning models. *Psychonomic bulletin & review*, **20**(2), 364-371.

Worthy, D., Pang, B., & Byrne, K. (2013). Decomposing the roles of perseveration and expected value representation in models of the iowa gambling task. *Frontiers in psychology*, **4**(640).

Yu, A., & Dayan, P. (2005). Uncertainty, neuromodulation, and attention. *Neuron*, **46**(4), 681-692.

Multi-Preference Actor Critic

Ishan Durugkar
Department of Computer Science
University of Texas at Austin
Austin, TX 78712, USA
ishand@cs.utexas.edu

Matthew Hausknecht & Adith Swaminathan & Patrick MacAlpine
Microsoft Research
Redmond, WA
{matthew.hausknecht, adswamin, Patrick.MacAlpine}@microsoft.com

Abstract

Policy gradient algorithms typically combine discounted future rewards and an estimated value function, to compute the direction and magnitude of parameter updates. However, for most Reinforcement Learning tasks, humans can provide additional insight to constrain the policy learning process. We introduce a general method to incorporate multiple different types of feedback into a single policy gradient loss. In our formulation, the Multi-Preference Actor Critic (M-PAC), these different types of feedback are implemented as constraints on the policy. We use a Lagrangian relaxation to approximately enforce these constraints using gradient descent while learning a policy that maximizes rewards. We also show how commonly used preferences can be incorporated into this framework. Experiments in Atari and the Pendulum domain verify that constraints are being respected and in many cases accelerate the learning process.

Keywords: reinforcement learning; actor critic; human preferences; constrained optimization

1 Introduction

We examine how to incorporate human preferences into policy gradient reinforcement learning algorithms to achieve higher performance in fewer environment interactions. Many existing papers have studied how human preferences can be incorporated into reinforcement learning: Human expert demonstrations, one of the more direct expressions of human preference, have been incorporated through a behavior-cloning pre-training phase or by mixing demonstrations with episodic experiences during updates [Hester et al., 2018, Nair et al., 2017]. When an expert is available to provide on-policy feedback, methods such as Dagger [Ross et al., 2011] and Aggrevate [Ross and Bagnell, 2014] query the expert to gain access to on-policy demonstrations, reducing the problem of covariate shift. Inverse Reinforcement Learning (IRL) attempts to use demonstrations to infer the demonstrator's reward function [Abbeel and Ng, 2004, Ziebart et al., 2008, Ho and Ermon, 2016]. IRL methods are particularly useful in tasks in which there are no explicit rewards and only expert demonstrations are available.

In certain domains, it is difficult for humans to provide direct demonstrations. Therefore a number of alternate ways of specifying human preferences have been explored: The TAMER framework [Knox and Stone, 2009] uses human feedback as an estimate of the value function. COACH [MacGlashan et al., 2017] shows that positive and negative feedback signals provided by humans during the course of an episode can be used to learn advantages over actions. Christiano et al. [2017] show that it is possible to learn complex behavior in environments using only a reward function inferred from asking humans to repeatedly choose between two potential policies.

Other human preferences are encoded as a part of the agent's loss function. For example, maximum-entropy reinforcement learning [Ziebart et al., 2008, Haarnoja et al., 2017] reflects the intuition that a policy should exhibit as much randomness as possible while maximizing rewards. This preference for greater entropy is expressed as a regularization term applied to the agent's objective function. Similarly, trust region methods [Schulman et al., 2015] enforce the preference that the learning agent's policy should not change drastically between updates.

Motivated by the variety of human preferences and feedback modalities, we construct a unifying architecture for learning from diverse human preferences. Multi-Preference Actor Critic (M-PAC) uses a single-actor network paired with multiple critic networks, where each source of preference feedback is encoded by a different critic. We formulate a constrained optimization problem in which each critic represents a soft constraint applied to the actor's policy, enforcing the corresponding human preference. This formulation allows us to use a Langrangian relaxation to automatically and dynamically learn the relative weighting of each preference. For example, in the early stages of the learning process, the agent may place strong emphasis on the critic encouraging similarity to human demonstrations, but later in the learning process, switch emphasis to ensure safe policy updates.

We conduct experiments combining four types of preferences: entropy regularization, safe policy updates, behavior cloning of expert demonstrations, and GAIL. Our experiments demonstrate that incorporating these preferences as critics in a constrained optimization framework allows faster learning and higher eventual performance. Furthermore, using this framework it is possible to incorporate other forms of human feedback in a straightforward manner.

2 Background

In this paper, we look at the setting of a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \mu, \gamma \rangle$ defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a reward function $r(s, a, s')$ and transition probabilities $\mathcal{P}(s, a, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$, where $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. μ is the initial state distribution of the MDP and $\gamma \in [0, 1)$ is the discount factor.

A policy $\pi(a|s)$ maps states to a probability distribution over actions. The discounted value of starting in a particular state at time t and then following policy π is given by

$$V_\pi(s) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s \right] \tag{1}$$

The advantage of taking an action a_t in state s_t can be considered as the additional value that the agent would get if it took action a_t and then followed policy π from state s_{t+1} over just following policy π in state s_t.

$$A_\pi(s_t, a_t) := r(s_t, a_t, s_{t+1}) + \gamma V_\pi(s_{t+1}) - V_\pi(s_t) \tag{2}$$

The aim of training a reinforcement learning agent is to find a policy that can maximize the agent's value over all states.

$$\pi^* := \arg\max_\pi \mathbb{E}_{s \sim \mu} V_\pi(s) \tag{3}$$

This search can be done using gradient descent by minimizing the loss $L = \mathbb{E}_\pi [-A_\pi(s, a)]$. Since $A_\pi(s, a)$ cannot be directly optimized in closed form, we typically use the policy gradient trick whereby the policy gradient is $\nabla_\theta L =$

$-\mathbb{E}_{\pi_\theta} A_{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)$. In practice, the A2C algorithm also adds an entropy term (H) to the loss, to prevent early convergence to a sub-optimal policy.

$$L = \mathbb{E}_{\pi_\theta} \left[-A_{\pi_\theta}(s,a) - \beta H(\pi_\theta(a|s)) \right] \tag{4}$$

3 Multi-Preference Actor Critic

Consider K possible preferences. We define preference $c_k(\pi)$ as a function mapping inputs π to $\mathbb{R}^+$. This preference metric measures the amount by which the preference is violated, and is 0 if the preference is perfectly satisfied. As a concrete example, consider a preference on agent behavior expressed through demonstrations $(s_1, a_1, s_2, a_2, \dots)$. We can do behavior cloning to learn a policy π_{bc} using this demonstration data. A preference $c_k(\pi)$ can then simply be the KL-divergence between π and π_{bc}: $c_k(\pi) = \mathbb{E}_\pi \log \pi(s,a) - \log \pi_{bc}(s,a)$. All the preferences we study in this paper can be viewed as expectations over samples drawn from π. As shorthand, we express these preferences as $c_k(\pi) = \mathbb{E}_\pi d_k(\pi, s, a)$; in the example above, $d_k(\pi, s, a) = \log \pi(s,a) - \log \pi_{bc}(s,a)$.

When we consider incorporating the above preferences into our policy search, observe that each c_k can be naturally interpreted as a critic in actor-critic architectures. So, one possibility is to add the preferences as additional costs incurred by the policy. The different preferences can then be folded into a single reward function by weighting each cost with a hyper-parameter.

$$L = \mathbb{E}_\pi \left[-A_\pi(s,a) + \sum_k \lambda_k d_k(\pi, s, a)) \right] \tag{5}$$

This is the approach we see in Kang et al. [2018], Gao et al. [2018], Hester et al. [2018], Nair et al. [2017]. It can be difficult to find the right hyperparameter values to weigh these preferences against each other. Moreover, the relative usefulness of individual preferences might change as the agent's proficiency increases.

A technique to incorporate varied preferences into the policy learning procedure that can weigh preferences in a principled manner is required. Consider instead a policy search procedure that is done only in the space of policies that satisfy the preferences. This leads to a constrained formulation of Equation 3.

Consider again the preference metric $c_k(\pi)$. We can specify how much the policy can stray from this preference by setting a threshold l_k. The search for the optimal policy can then be written as

$$\pi^* := \arg\max_\pi \mathbb{E}_{s\sim\mu} V_\pi(s) \tag{6}$$

$$\text{s.t. } \forall k \left[\mathbb{E}_{s\sim\mu} \sum_a d_k(\pi, s, a) \right] \leq l_k \tag{7}$$

If our preferences are sufficiently diverse, the set of policies that satisfies the above constraints will be much smaller than the set of all policies we were searching over when we had only the environmental returns to guide us.

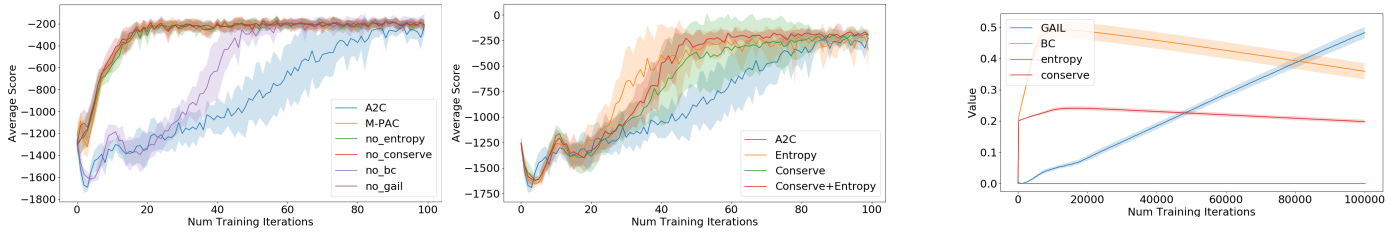
We can now turn to Lagrangian relaxation of these constraints so that they are no longer hard constraints and furthermore, we can use policy gradient to find a feasible policy. If the agent policy is a function with parameters θ, Lagrangian relaxation with parameters $\lambda_k \in \mathbb{R}^+$ on the constraints leads to the following saddle point problem.

$$\min_\theta \max_\lambda \mathbb{E}_{\pi_\theta} \left[-A_{\pi_\theta}(s,a) + \sum_k \lambda_k \left(d_k(\pi_\theta, s, a) - l_k \right) \right]. \tag{8}$$

When we use policy gradients or any other stochastic gradient method to optimize this saddle point formulation, the λ_k weight is increased if the preference is violated beyond our threshold. It decreases to 0 if the preference metric stays within that threshold. Policy gradient simultaneously updates θ to minimize the joint objective. Equation 8 looks remarkably like Equation 5, but also offers a principled way to adjust the weighting on the preferences.

4 Examples of Preferences

We now consider how the preferences that we discussed in Section 1 can be incorporated in the M-PAC framework we set up in Section 3. To incorporate preferences we convert a preference into the form of a function (d_k) that maps π, s, a to $\mathbb{R}$. We show below that this conversion is fairly straightforward for all the preferences we have considered.



(a) Demonstrations (b) No Demonstrations (c) Changing λ

Figure 1: We assess the relative importance of each preference, and how lambdas change over time. In Figure 1a, we utilize demonstrations provided by a policy learned by A2C to learn the reference policy and GAIL reward. In Figure 1b we compare the performance when no demonstrations are available. We see that both entropy and conservative update preferences accelerate learning compared to vanilla A2C. Figure 1c shows change of λ values for different preferences.

- **Entropy**:
  A high entropy policy is a common preference [Haarnoja et al., 2017] that is usually enforced as a regularization by means of a surrogate loss. We can present a high entropy policy as a preference, and define the entropy preference $d_{entropy}(\pi, s)$ as

$$d_{entropy}(\pi, s) = KL(\pi(s)||q) \tag{9}$$

$$\forall a \in \mathcal{A} : q(a) = \frac{1}{\|\mathcal{A}\|}. \tag{10}$$

- **Conservative Updates**:
  We introduce a preference of staying close to a previous policy, updated slowly closer to the current one. Let θ' be the parameters of this older policy. The conservative policy preference is given by

$$d_{conserve}(\pi_\theta, s) = KL(\pi_\theta(s)||\pi_{\theta'}(s)). \tag{11}$$

- **Reference Policy**:
  A reference policy can be given a priori, or learnt from expert demonstrations by Behavior Cloning.

$$d_{reference}(\pi_\theta, s) = KL(\pi_\theta(s)||\pi_{ref}(s)). \tag{12}$$

- **Inverse RL**:
  Preferences can also be expressed as a cost-to-go function. Here we consider a reward function deduced from demonstrations using GAIL [Ho and Ermon, 2016]. We learn a value function $V_{gail}(s)$ and use it to calculate the advantage $A_{gail}(s, a)$ (using Eqn (2)) of taking an action. The GAIL preference is then defined as

$$d_{gail}(\pi_\theta, s, a) = -\log \pi_\theta(a|s) A_{gail}(s, a). \tag{13}$$

5 Experiments

We instantiated M-PAC using Advantage Actor Critic (A2C) as the base policy gradient learning algorithm upon which the preferences in Section 4 were incorporated. To show that M-PAC is capable of incorporating various types of preferences, we compare M-PAC and A2C on the Pendulum domain. An ablation analysis is also performed to analyze the effect of the different preferences in this domain. Refer to Figure 1 for the results.

We save demonstrations by running a pre-trained policy trained using A2C that gets an average score of -195. These demonstrations are then used for behavior cloning and for learning the GAIL reward. Both the A2C and M-PAC policies are learned through a multi-layer perceptron with 2 layers of 512 units each. Figures are plotted by taking an average of 10 independent runs with separate seeds.

From Figure 1a, we can see that with enough data, both GAIL and Behavior Cloning can provide enough guidance for the agent to explore and reach the optimal behavior faster than A2C. Learning from a reference policy directly is very beneficial initially, while GAIL provides a more indirect exploration signal due to its adversarial learning procedure.

To compare the benefits of the preferences for conservative updates and a high entropy policy, we consider Figure 1b. Both these preferences individually and in tandem help the policy learn faster than A2C. Although A2C already employs entropy regularization, we hypothesize that our entropy constraint does better because it affects the policy learning only when the policy violates the determinism threshold.

Another important aspect of M-PAC is the λ parameters and how they change with respect to violated constraints. Figure 1c compares the λ parameters for the different preferences considered when we are using all the preferences (M-PAC from

Game	A2C	PPO	M-PAC
MsPacman	1686.1	2096.5	**2495.22**
BeamRider	3031.7	1590.0	**3157.36**
Breakout	303.0	274.8	**326.675**
Pong	19.7	**20.7**	20.41
Seaquest	**1714.3**	1204.5	908.33
SpaceInvaders	744.5	**942.5**	600
Qbert	5879.25	**14293.3**	3769.37

Table 1: **Comparison on ALE** Using only conservative updates and entropy regularization, M-PAC outperforms A2C and PPO on three games, ties on one, and performs worse on three.

Figure 1a). Here we see that the λ associated with behavior cloned reference policy preference (BC) and conservative update preference (conserve) shoots up to a value where they can control these values according to the threshold we set. λ associated with the GAIL advantages keeps climbing steadily as the advantages keep providing signal to the policy, and is eventually weighted more than the behavior cloned reference policy. In all this while, the policy entropy does not stray far enough away from a high entropy reference to cause its λ to increase from 0.

Since M-PAC provides a novel way to enforce trust-region and entropy constraints, we perform a parameter sweep on seven well-known Atari games and compare to A2C (with entropy regularization) and PPO (with trust-region constraints). The results in Table 1 show that, even without any additional human feedback, there are some environments where M-PAC's way of enforcing these constraints can yield a substantially better policy and merits further study.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

Yang Gao, Ji Lin, Fisher Yu, Sergey Levine, Trevor Darrell, et al. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning*, 2017.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John Agapiou, Joel Z Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations. In *Annual Meeting of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems 29*, pages 4565–4573. 2016.

Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pages 2474–2483, 2018.

W. Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The TAMER framework. In *The Fifth International Conference on Knowledge Capture*, September 2009.

James MacGlashan, Mark K. Ho, Robert Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E. Taylor, and Michael L. Littman. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2285–2294, 2017.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *arXiv preprint arXiv:1709.10089*, 2017.

Stéphane Ross and J. Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *CoRR*, abs/1406.5979, 2014.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 15 of *JMLR Proceedings*, pages 627–635, 2011.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1433–1438. AAAI Press, 2008.

Joint Goal and Constraint Inference using Bayesian Nonparametric Inverse Reinforcement Learning

Daehyung Park
daehyung@csail.mit.edu

Michael Noseworthy
mnosew@mit.edu

Rohan Paul
rohanp@csail.mit.edu

Subhro Roy
subhro@csail.mit.edu

Nicholas Roy
nickroy@csail.mit.edu

Computer Science and AI Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

Inverse Reinforcement Learning (IRL) aims to recover an unknown reward function from expert demonstrations of a task. Often, the reward function fails to capture a complex behavior (e.g., a condition or a constraint) due to the simple structure of the global reward function. We introduce an algorithm, *Constraint-based Bayesian Non-Parametric Inverse Reinforcement Learning* (CBN-IRL), that instead represents a task as a sequence of subtasks, each consisting of a goal and set of constraints, by partitioning a single demonstration into individual trajectory segments. CBN-IRL is able to find locally consistent constraints and adapt the number of subtasks according to the complexity of the demonstration using a computationally efficient inference process. We evaluate the proposed framework on two-dimensional simulation environments. The results show our framework outperforms state-of-the-art IRL on a complex demonstration. We also show we can adapt the learned subgoals and constraints to randomized test environments given a single demonstration.

Keywords: Learning from Demonstration, Inverse Reinforcement Learning, Markov Decision Process

Acknowledgements

We gratefully acknowledge funding support in part by the U.S. Army Research Laboratory under the Robotics Collaborative Technology Alliance (RCTA) Program, Lockheed Martin Co., and the Toyota Research Institute Award LP-C000765-SR.

1 Introduction

Inverse Reinforcement Learning (IRL) aims to recover an unknown reward function from an expert's demonstrations of a task. Often, this reward function will need to represent complex behaviour which is a function of both the final state and local properties of the demonstrations. For example, in a peg-in-hole assembly task, the peg should be vertically aligned when placed into the hole but while moving through free-space, it is not necessary to keep the peg upright. Previous work has approached the problem of learning a complex reward function by composing a set of task-relevant features. As conventional approaches (which represent a task as a single reward function) are often data-intensive, Michini and How [3] proposed *Bayesian Nonparametric IRL* (BN-IRL) as a way to recover reward functions from considerably less data by decomposing a task into a sequence of simpler tasks, each consisting of a simple 0-1 reward. The approach segments each demonstration into an *a priori* unknown number of subtasks and infers a subgoal per segment. This approach provides superior performance in IRL tasks when the state space is large and the demonstration dataset is small. Specifically, the model imposes a *Chinese Restaurant Process* (CRP) prior [5] on the partitions of the demonstration. However, to capture complex behaviour, BN-IRL requires a large number of subtasks, increasing the computational complexity of inference. In this work, we propose *Constraint-Based* BN-IRL (CBN-IRL) that captures the behaviour of each subtask with both a subgoal and set of constraints. The constraints allow more complex behaviour to be encoded in each segment and reduces the number of subtasks needed to represent the entire task. We provide an efficient algorithm to extend the BN-IRL Gibbs sampling to handle constraints and compare its performance with other IRL algorithms in 2D environments.

2 Background : BN-IRL

IRL aims to recover the reward function for a *Markov Decision Process* (MDP) defined by the 5-tuple, (S, A, T, R, γ). S, A, and $\gamma \leq 1$ are the state-space, action-space, and a discount factor. T is a stochastic transition function and R a reward function. The agent is provided with a set of demonstrations that are assumed to have come from an optimal policy. We represent a demonstration as a set of observations $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, ...\}$ where each observation is a state action pair $\mathcal{O}_i = (s_i, a_i)$.

BN-IRL uses an infinite mixture model to segment the demonstration into a set of partitions, each represented as a simple reward function given by a subgoal $R_{\mathbf{g}}(s_i) = \mathbb{1}(s_i = g)$. For each timestep in the demonstration, a latent assignment-variable $z_i \in \mathbb{N}$ is introduced to associate the observed state-action at each time step i in the demonstration with a specific partition and corresponding subgoal. The distribution over assignments from observed $\mathcal{O}_i$ to partition z_i is modeled by a CRP to handle an unknown number of subgoals.

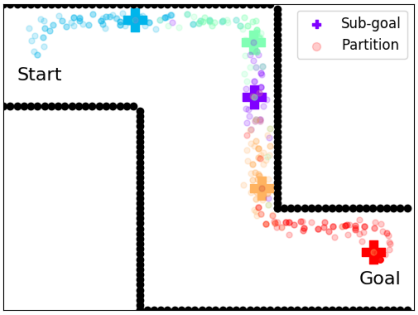


Figure 1: A 2D navigation environment. Plus and circle markers show subgoals and assignment-variables estimated from BN-IRL. Black blocks represent walls.

The goal of BN-IRL is to infer the most likely set of partitions and corresponding subgoals, $\mathbf{g}$, and assignment-variables, $\mathbf{z}$, given a demonstration. The joint distribution of observations, goals, and partitions is:

$$p(\mathcal{O}, \mathbf{g}, \mathbf{z}) = \prod_{i=1}^{N} \underbrace{p(\mathcal{O}_i|g_{z_i})}_{\text{likelihood}} \underbrace{p(z_i|z_{-i}, \eta)}_{\text{CRP}} \prod_{j=1}^{J_i} \underbrace{p(g_j)}_{\text{prior}}, \tag{1}$$

where J_i is the number of observations assigned in the ith partition. The likelihood represents how likely an agent behaving optimally with respect to the given goal would have generated the observation. The conditional distribution of the assignment-variables is represented as:

$$p(z_i|z_{-i}, \eta) = \begin{cases} \frac{m_i}{n-1+\eta} & \text{if } 1 \leq i \leq K \\ \frac{\eta}{n-1+\eta} & \text{otherwise,} \end{cases}$$

where n is the total number of observations, and m_i is the number of observations in partition i. η is the concentration parameter of the CRP and K is the current number of partitions. The CRP prior with a high concentration hyper-parameter η allows infinite partitions and hence grows the model complexity with respect to new observations. Inference is performed using uncollapsed Gibbs sampling, which is discussed further in Section 3.

3 Constraint-based BN-IRL (CBN-IRL)

CBN-IRL extends BN-IRL by assuming that the trajectory in each partition can be modelled as a subgoal, g_i, and active constraints, c_i. Each constraint places strict restrictions on which states an agent can visit. The problem therefore is to infer the segmentation of the demonstrated trajectory into smaller partitions and for each partition, infer the subgoal and constraints, that when given as input to a planner, result in a trajectory that maximizes the likelihood of the observation.

A state is represented by a set of features, $\mathbf{f}$, and each constraint operates on these features to denote valid and invalid regions of the state space. In this work, we use a set of handcrafted features calculated using the state of the agent with a world model (the world model represents environment variables such as the location of an obstacle). In particular, we use coordinate-free continuous features such as *minimum distance from a goal*, *minimum distance from obstacles*, etc.

Each constraint c is a conjunction of simple inequality constraints, which ensures all constraints are satisfied: $c = c^1 c^2 \cdots c^K$, where K is the number of features used to define c. Each inequality constraint, c^i, is computed from the ith feature and used to check if a feature, $f^i \in \mathbf{f}$ falls within a bound $\zeta \in \mathbb{R}$ of some given expected feature value $\hat{f}^i$:

$$c^i :\mid f^i - \hat{f}^i \mid < \zeta \rightarrow \{1, 0\}. \tag{2}$$

In practice, we estimate $\hat{f}^i$ as the feature-mean of all the states belonging to the same subtask.

3.1 Generative Model

A trajectory can be seen as the observations of a generative process where the subgoals and constraints are latent variables $\boldsymbol{\theta} = \{(g_1, c_1), ..., (g_N, c_N)\}$. In this work, we assume subgoals $\mathbf{g}$ are a subset of the states observed in an expert demonstration and that the demonstration satisfies all task constraints. We further assume the demonstration is the only optimal path from a start state to a goal state. Then, the joint probability distribution of a new state-action trajectory of length l, $\mathbf{x} = \{s_0, a_0, s_1, a_1, ..., s_l, a_l\}$, and a demonstration, $\mathcal{O}$, is:

$$p(\mathbf{x}, \mathcal{O}) = p(\mathbf{x}|\mathcal{O})p(\mathcal{O}) = \int_{\boldsymbol{\theta}, \mathbf{z}} \underbrace{p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{z})}_{\text{Likelihood}} \quad \underbrace{p(\boldsymbol{\theta}, \mathbf{z}|\mathcal{O})}_{\substack{\text{subgoals\&} \\ \text{-constraints inference}}} \quad p(\mathcal{O}). \tag{3}$$

3.2 Inference

We now describe how to estimate the posterior over latent variables, $(\boldsymbol{\theta}, \mathbf{z})$, given a demonstration, which can be used to get the MAP estimate of the likelihood of the subgoals, active constraints for each partition, and assignment variables associated with the demonstration:

$$\boldsymbol{\theta}^*, \mathbf{z}^* = \arg\max_{\boldsymbol{\theta}, \mathbf{z}} p(\boldsymbol{\theta}, \mathbf{z}|\mathcal{O}). \tag{4}$$

Since the space of $(\boldsymbol{\theta}, \mathbf{z})$ is larger than BN-IRLs latent space (g, z), exact inference is intractable. To approximate it we use *Block Gibbs Sampling* which iteratively samples $\mathbf{g}$, $\mathbf{c}$, and $\mathbf{z}$ by conditioning on the others (see Algorithm 1). After running Gibbs sampling, the empirical mode of the samples $\boldsymbol{\theta}^{1:t}$ and $\mathbf{z}^{1:t}$ will have converged to values that approximate the true values of the latent parameters assuming the generative process of Eq. 3 (see Lemma 6.1). We describe the Gibbs sampling updates of $\mathbf{z}$ and $\boldsymbol{\theta}$ below.

3.2.1 Partitions

The conditional probability distribution of a partition z_i given the other partitions, z_{-i}, observations, $\mathcal{O}$, and goals/constraints, θ, is:

$$p(z_i|z_{-i}, \boldsymbol{\theta}, \mathcal{O}, \eta) \propto \underbrace{p(\mathcal{O}_{z_i}|\theta_{z_i})}_{\text{likelihood}} \underbrace{p(z_i|z_{-i}, \eta)}_{\text{CRP prior}}, \tag{5}$$

where $\mathcal{O}_{z_i}$ are the observations assigned to partition z_i and depend on the subgoal and constraint pair for that partition, θ_{z_i}.

The likelihood represents the probability that an agent acting optimally under the current constraints and subgoal generates the given observations similar to [3]:

$$p(\mathcal{O}_{z_i}|\theta_{z_i}) = p(a_i|s_i, g_{z_i}, c_{z_i}) = \frac{e^{\alpha Q^*(s_i, a_i, R_{g_{z_i}}, c_{z_i})}}{\sum e^{\alpha Q^*(s_i, a_i, R_{g_{z_i}}, c_{z_i})}}, \tag{6}$$

where α is a parameter representing the degree of confidence and Q^* is the optimal Q-function from the constrained MDP/$< T_c, R_g >$. R_g is a reward function, $R_g(s_i) = \mathbb{1}(s_i = g)$. To represent constraints, $\mathbf{c}$, we define a constraint-based transition probability function, T_c, that disallows transitions into invalid states (i.e., a truncated transition function),

$$T_c(s, a, s') = \begin{cases} 0 & \text{if } c(s') = \textit{false} \\ T(s, a, s')/\Sigma_{s''} T(s, a, s'') & \text{otherwise,} \end{cases} \tag{7}$$

Algorithm 1: CBN-IRL

Input: $\mathcal{O} = \{\mathbf{x}^d\} = \{(x_1, a_1, x_2, ...)\}, \boldsymbol{\theta}^{(0)}, \mathbf{z}^{(0)}$
Pre-computation of Q^*-functions given potential $\boldsymbol{\theta} = (\mathbf{g}, \mathbf{c})$
for $t \leftarrow 1$ ***to*** *MaxIter* **do**
  for $i \leftarrow 1$ **to** $|\mathbf{z}|$: Draw $\theta_i^{(t)} \sim p(\theta_i \mid \mathbf{z}, \mathcal{O})$
  for $i \leftarrow 1$ **to** $|\mathcal{O}|$: Draw $z_i^{(t)} \sim p(z_i|z_{-i}, \boldsymbol{\theta}, \mathcal{O}, \eta)$
end
return $\boldsymbol{\theta}^{(1:t)}, \mathbf{z}^{(1:t)}$

where $T(s, a, s')$ is a transition probability function, $\sum_{s'} T(s, a, s') = 1$, and $s'' \in \{s | s \in S, c(s) = true\}$. To expedite the learning process, we pre-compute Q^* for each combination of g_i and c_i. We can also use constrained Markov decision process (CMDP) [1] or its latest variants to obtain the Q-function given complex constraints.

3.2.2 Subgoals and Sub-Constraints

We define the conditional distribution of a subgoal and constraint pair $\theta_j = (g_j, c_j)$ given a specific partition $\mathbf{z}$ as:

$$p(\theta_j|\mathbf{z}, \mathcal{O}) \propto p(\mathcal{O}_j|\theta_j, \mathbf{z}, \mathcal{O}_{-j}) p(\theta_j|\mathbf{z}, \mathcal{O}_{-z_j}) = \sum_{i \in I_j} \underbrace{p(\mathcal{O}_i|\theta_{z_i})}_{\text{likelihood}} \underbrace{p(\theta_j)}_{\text{prior}}, \tag{8}$$

where $I_j = \{i : z_i = j\}$. The inference of the joint subgoals and constraints is computationally expensive due to the combinatorially large space of constraints. To simplify this, we sample goals and constraints independently assuming conditional independence. Specifically, the goal is sampled from the set of observations assigned to the given partition. In general, a constraint could consist of simple constraints for any subset of features. To sample constraints, we limit the class of constraints considered to either be a conjunction of *every* feature or the empty set representing no constraints: $c_j = \prod_{\{k | \hat{f}^k \in \hat{\mathbf{f}}_{\text{free}}\}} c_j^k$ or $\emptyset$. The features used for these constraints are goal-free $\mathbf{f}_{\text{free}}$ (e.g., *distance from objects*) which allow constraints that can generalize to new goals.

The likelihood is calculated as in the previous section. As the sampling process requires re-estimation of the valid feature range, $\hat{f}_j^k$, used to compute constraints based on partition assignments at each update, it is hard to pre-compute Q^* as previously done. To resolve these issues, we pre-compute a list of potential feature-means, $F = \{\hat{f}^{k,1}, \hat{f}^{k,2}, ..., \hat{f}^{k,M}\}$, from a discretized feature space, where M is the resolution and the extent of the space $(\hat{f}^{k,1}, \hat{f}^{k,M})$ is obtained from the minimum and maximum value of features in the demonstration $\mathcal{O}$, respectively. CBN-IRL then pre-computes Q per potential constraint defined from each feature-mean in F. During the sampling, CBN-IRL selects a pre-computed Q from the closest pre-computed feature-mean as Q^*.

3.3 Prediction

CBN-IRL predicts a current partition z of a current state s and then predicts the maximum likelihood of action a given the current z and the empirical mode of samples $(\boldsymbol{\theta}^*, \mathbf{z}^*)$: $z^*, a^* = \arg\max_{z_i, a} p(a|s, g_{z_i}, c_{z_i})$.

4 Evaluation

We evaluate CBN-IRL and 4 baseline methods with respect to their ability to reproduce the demonstrated trajectory both within the same 2D environment and in novel 2D environments. Performance is measured by the distance between a generated and demonstrated trajectory. Here we use *Dynamic Time Warping* (DTW) [2] to measure the distance between the trajectories. The underlying discrete state space $\mathcal{S}$ of the MDP is given by a random sampling of positions in the 2D environment.

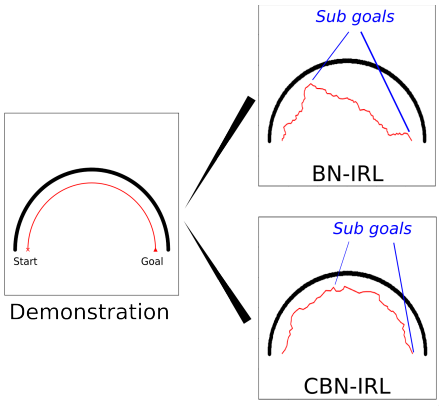


Figure 2: Comparison between BN- and CBN-IRL given a semi-circular path demonstration. Red markers show start and goal locations, respectively. Black blocks represent walls that an agent cannot pass through.

We first show the benefit of constraints by reproducing a non-linear trajectory in a 2D semi-circular obstacle environment. Fig. 2 shows a manually-drawn expert demonstration (red line) that keeps a certain distance from the black obstacles. BN-IRL was able to infer two subgoals but failed to keep a consistent distance from the obstacle. By increasing the number of subgoals (by changing η), BN-IRL was able to reduce the loss (DTW distance) from the expert demonstration (see Fig. 3 **Left**). On the other hand, CBN-IRL also found two subgoals but was able to produce a demonstration-like trajectory by limiting its actions to stay close to the obstacle. Fig. 3 **Left** shows CBN-IRL resulted in the lowest loss from the demonstration. Conventional IRL methods, such as MaxEnt [7] and Deep-MaxEnt IRL [6], resulted in significantly lower performance due to the single demonstration and simple features. These baselines used the same two features as in CBN-IRL and the distance from the center of the circle.

We now show the generalization performance by comparing with BN-IRL in 1,000 randomized sinusoidal passages $(= h \cdot \sin(w\theta + \theta_0))$, where we uniform-randomly sampled (h, w, θ_0) in $0.5 \leq h \leq 3$, $0.5 \leq w \leq 3$, and $-\pi \leq \theta_0 \leq \pi$. Each method was trained on just one demonstrated passage. We generate an optimal path that keeps a certain distance between

two passage walls to measure performance against. Fig. 3 **Right** shows CBN-IRL resulted in lower average loss than BN-IRL. Regardless of the number of subgoals, CBN-IRL was able to produce consistent results using the feature-based constraints. BN-IRL resulted in the highest loss given $\eta = 0.1$ which only recovered 2-4 partitions. The simple reward structure from BN-IRL with few subgoals cannot draw sinusoidal curves. This emphasizes the fact that the subgoal-only model requires many more subgoals to reproduce complex behavior.

5 Conclusion and Future Work

We introduced a *Constraint-based* Bayesian Non-parametric Inverse Reinforcement Learning framework (CBN-IRL) which partitions a demonstration into a series of tasks each represented by a subgoal and set of sub-constraints. Our approach increases the expressivity while maintaining the data efficiency of BN-IRL. Like BN-IRL, our method does not require the number of subgoals and constraints to be pre-specified, and the addition of constraints results in a more compact and therefore computationally efficient approach overall. By adding constraints, our method was able to significantly reduce the loss from a demonstration and generalize to new environments without increasing the number of subgoals. This work will be extended to high-dimensional robotic assembly tasks where task constraints are required to successfully complete assigned tasks.

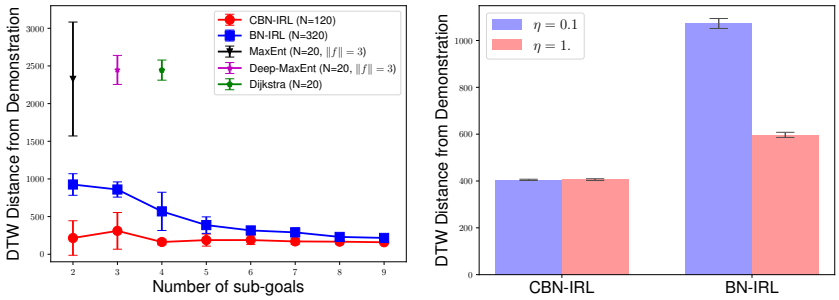


Figure 3: Evaluation in a 2D environment. **Left:** DTW performance comparison with four baseline methods given the semi-circular path demonstration in Fig. 2 **Right:** Generalization performance in 1,000 randomized sinusoidal path. η is the concentration hyperparameter in the CRP. The black caps show standard error.

6 Appendix: Effectiveness of Constraints

As in BN-IRL [3], we let $\mathcal{L}(\cdot,\cdot)$ measure the distance between the estimated and true latent variables, $(\hat{\mathbf{z}}, \hat{\mathbf{g}}, \hat{\mathbf{c}})$.

Lemma 6.1. *Assuming an expert state-action trajectory* $\mathbf{x}^*$ *is generated from the model defined by Eq. (3), the expected loss,* $\mathcal{L}((\boldsymbol{\theta}, \mathbf{z}), (\hat{\boldsymbol{\theta}}, \hat{\mathbf{z}}))$*, between the predicted and true underlying latent variables, is minimized by the empirical mode of the samples* $(\boldsymbol{\theta}, \mathbf{z})$ *as the number of iterations* $t \to \infty$.

Proof. The expected loss between the true and inferred latent variables in the sub-goal-only model, BN-IRL, is minimized by the empirical mode of the samples $(\mathbf{g}, \mathbf{z})$ as the number of Gibbs sampling iterations $t \to \infty$ according to Theorem 1 in [3]. Likewise, given a set of valid constraints, $\mathbf{c}$, the Markov chain in the block Gibbs sampling for CBN-IRL also converges to the true mode of values, $\boldsymbol{\theta}$ and $\mathbf{z}$, from the true posterior $p(\boldsymbol{\theta}, \mathbf{z}|\mathcal{O})$. The sufficient condition for ergodicity of the Markov chain [4] is the conditional probabilities defined by Eq. (5) and (8) are non-zero for all states and constraints. In Eq. (5), both the likelihood of the observation (i.e., expert trajectory) and the CRP are always positive by the assumption of valid and discrete constraint sets. In Eq. (8), the likelihood is greater than zero and the prior is a positive constant. Thus, by the strong law of large numbers, as the number of Gibbs sampling iterations $t \to \infty$, the empirical mode of the samples is

$$(\boldsymbol{\theta}, \mathbf{z}) = \underset{\boldsymbol{\theta}^{(1:t)}, \mathbf{z}^{(1:t)}}{\arg\max}\ p(\boldsymbol{\theta}, \mathbf{z}|\mathcal{O}). \tag{9}$$

□

References

[1] Eitan Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[2] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[3] Bernard Michini and Jonathan P How. Bayesian nonparametric inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer, 2012.

[4] Radford M Neal. Probabilistic inference using markov chain monte carlo methods. *Department of Computer Science, University of Toronto Toronto, ON, Canada*, 1993.

[5] Jim Pitman et al. Combinatorial stochastic processes. Technical report, Dept. Statistics, UC Berkeley, 2002.

[6] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[7] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, volume 8, pages 1433–1438, 2008.

ProtoGE: Prototype Goal Encodings for Multi-goal Reinforcement Learning

Silviu Pitis*
University of Toronto, Vector Institute
Toronto, ON, Canada
spitis@cs.toronto.edu

Harris Chan*
University of Toronto, Vector Institute
Toronto, ON, Canada
hchan@cs.toronto.edu

Jimmy Ba
University of Toronto, Vector Institute
Toronto, ON, Canada
jba@cs.toronto.edu

Abstract

Current approaches to multi-goal reinforcement learning train the agent directly on the desired goal space. When goals are sparse, binary and coarsely defined, with each goal representing a set of states, this has at least two downsides. First, transitions between different goals may be sparse, making it difficult for the agent to obtain useful control signals, even using Hindsight Experience Replay [1]. Second, having trained only on the desired goal representation, it is difficult to transfer learning to other goal spaces.

We propose the following simple idea: instead of training on the desired *coarse* goal space, substitute it with a *finer*—more specific—goal space, perhaps even the agent's state space (the "state-goal" space), and use Prototype Goal Encodings ("ProtoGE") to encode coarse goals as fine ones. This has several advantages. First, an agent trained on an appropriately fine goal space receives more descriptive control signals and can learn to accomplish goals in its desired goal space significantly faster. Second, finer goal representations are more flexible and allow for efficient transfer. The state-goal representation in particular, is *universal*: an agent trained on the state-goal space can potentially adapt to arbitrary goals, so long as a Protoge map is available. We provide empirical evidence for the above claims and establish a new state-of-the-art in standard multi-goal MuJoCo environments.

Keywords: multi-goal reinforcement learning, task specification, transfer learning, hindsight experience replay

1 Introduction, Background & Related Work

Humans can often accomplish specific goals more readily that general ones. Although more specific goals are, by definition, more challenging to accomplish than more general goals, evidence from management and educational sciences supports the idea that "specific, challenging goals lead to higher performance than easy goals" [9]. We find evidence of this same effect for reinforcement learning (RL) agents in multi-goal environments. Our work establishes a new state-of-the-art in standard multi-goal MuJoCo environments and suggests several novel research directions.

Multi-Goal Reinforcement Learning We consider the multi-goal RL setting, where an agent interacts with an environment and learns to accomplish different goals. The problem is described by a generalized Markov Decision Process (MDP) $\langle S, A, T, G \rangle$, where S, A, T, and G are the state space, action space, transition function and goal space, respectively [15, 16]. In the most general version of this problem each goal is a tuple $g = \langle R_g, \gamma_g \rangle$, where $R_g : S \to \mathbb{R}$ is a reward function and $\gamma_g \in [0, 1]$ is a discount factor [16], so that "solving" goal $g \in G$ amounts to finding an optimal policy in the classical MDP $\langle S, A, T, R_g, \gamma_g \rangle$. We focus on the sparse, binary reward case where each goal g corresponds to a set of "success" states, $\mathcal{S}(g)$, with $R_g : S \to \{-1, 0\}$ and $R_g(s) = 0$ if and only if $s \in \mathcal{S}(g)$ [13]. In this setting, the agent must learn to achieve and maintain success.

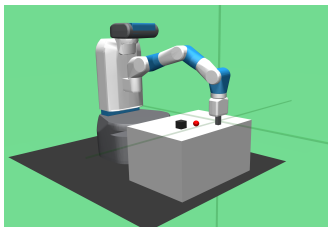


Fig. 1: In `Push`, the agent must push the black box onto the red target.

We use three multi-goal `Fetch` environments from OpenAIGym [3]: `Push`, `PickAndPlace`, and `Slide` (`v0`) [13]. In `Push` and `PickAndPlace`, the agent must use its gripper to move a box to the desired location (Figure 1). In `Slide`, the agent must hit a puck so it slides onto and stops in the desired location beyond the reach of the robot. A discount factor of $\gamma = 0.98$ is used and the environments reset (a new goal is sampled) every 50 steps. Goals are specified by a 3-dimensional vector of x, y and z coordinates. The goal is satisfied and reward of 0 is obtained so long as the box or puck is within an epsilon ball of the goal. An episode is a "success" only if the goal is satisfied on the final (50th) step.

Goal-Conditioned Actor-Critic Algorithms Many RL algorithms can be decomposed into actor (policy) and critic (value function). The DQN algorithm, for discrete action spaces, parameterizes both a greedy actor and a critic using the same deep neural network [10]. DDPG [7], the continuous action space equivalent of DQN, parameterizes actor and critic separately. Both DQN and DDPG are off-policy algorithms and use a replay buffer to store past experiences; this buffer is sampled from to train the actor and critic networks. To use DQN and DDPG in the multi-goal setting, we adopt the standard approach, which generalizes the actors (critics) to be functions of not only the state s (and action a), but also the goal g. A goal-conditioned critic is referred to as a GVF [16] or UVFA [14].

Hindsight Experience Replay An untrained agent acting in a sparse reward environment rarely achieves success, which makes standard training of goal-conditioned actors and critics difficult. Hindsight Experience Replay (HER) [1] accelerates learning by augmenting real experiences in the agent's replay buffer with fake "potential" goals. The intuition behind HER is that failures are informative: a failed attempt to reach g may have led to some other potential goal g'. By pretending that g' was the agent's true goal, the agent can learn something useful even from failed attempts.

When applying HER, one must choose an appropriate sampling strategy for potential goals. The previous best strategy was the `future` strategy, which chooses potential goals to correspond to future states visited along the same trajectory. Since `future` requires us to map visited states to goals that would have been achieved in those states, we must assume that "given a state s we can easily find a goal g which is satisfied in this state" [1]. Below, we propose a novel goal sampling strategy `futureactual`, which results in state of the art performance when used together with Protoge.

2 Protoge

Desired goals in `Fetch` are completely specified by the object's (box or puck) target location—the position of the gripper is irrelevant, as are other state variables. Such goals are coarse, in that each goal corresponds to a large number of states. Using a coarse goal specification during training has at least two downsides. First, as coarser goals have larger success state sets, "achieving" a goal provides relatively less control signal. An untrained agent acting in `Push`, for example, often fails to move the box at all. In this case, the `future` strategy samples only a single potential goal, which is satisfied by *all* states in the trajectory, and little is learned. Second, it is difficult for an agent to transfer its learning to another goal space: there is no natural way for a trained `Push` agent to transfer its knowledge in order to both (1) push the box to location A, and then (2) move the gripper to location B.

To address these difficulties, we propose to train the agent on a finer—more specific—goal space and use Prototype Goal Encodings (ProtoGE) to encode coarse goals as fine ones. Formally, we say that goal space A is **coarser** than goal space B if and only if there exists **Protoge map** $f : A \to B$ such that for each goal $a \in A$, the success state set $\mathcal{S}_B(f(a))$ of its

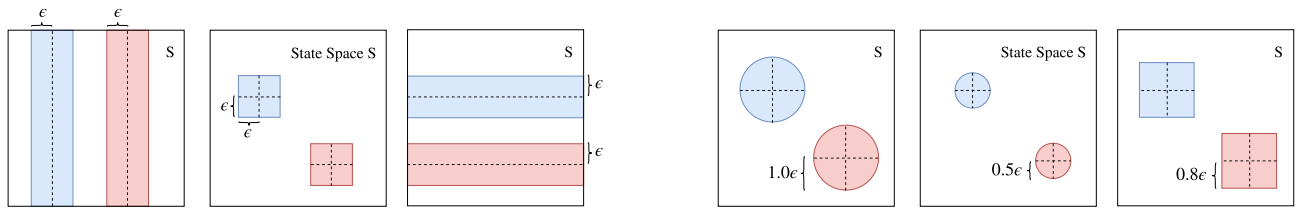


Fig. 2: Concept diagrams of *feature Protoges* (left) and *epsilon Protoges* (right). Each frame represents a different goal space over the same underlying state space, and the colored shapes inside represent success state sets for two example goals. On both the left and right, the middle goal space is finer than the adjacent spaces, which are incomparable. The goals (squares, small circles) in the middle spaces are valid Protoges for the corresponding goals (slices, large shapes) in the outer spaces.

Protoge, $f(a)$, is a subset of a's success state set $\mathcal{S}_A(a)$. If A is coarser than B, B is **finer** than A. If neither is coarser, A and B are **incomparable**. Below, we consider two types of Protoges: **feature Protoges**, which expand the dimensionality of the goal feature representation, and **epsilon Protoges**, which use tighter epsilon balls for determining goal achievement. See Figure 2 for conceptual diagrams of each. Note that any topological basis of S is finer than any goal space whose elements are open sets in S (an "open goal space"). In particular, if $S \subset \mathbb{R}^n$, the standard basis of epsilon balls about each state $s \in S$, $\mathcal{B}_\epsilon(s) = \{x \mid \|x - s\| < \epsilon\}$ where $\epsilon \in \mathbb{R}^+$, is finer than all open goal spaces. We call this the (ϵ-) **state-goal space**. Finally, note that two goal spaces A and B can both be finer than the other (as are, e.g., any two topological bases).

For example, we replace the 3-dimensional goals in `Push` with 6-dimensional goals indicating not only a target box position, but also a gripper position. Because any gripper position satisfies the original desired goal, we use a feature Protoge and require that the agent place the gripper *above* the target box location. An agent must achieve the original goal in order to achieve the Protoge. Using Protoges has at least two advantages, described below.

3 Accelerated Learning with Protoge

Although finer goals are, by definition, harder to accomplish than coarse ones, transitions between fine goals are less sparse and provide the agent with more control signal. When the gripper position is added to the agent's goal representation in the `Push` task, every movement achieves a new potential goal, even when the box remains unmoved, allowing the agent to learn how to control the gripper when using HER's `future` strategy. The intuition here is similar to that of auxiliary tasks [6] and GVFs [16]: learning to control things—even if not directly connected to the agent's primary goal—results in better overall control and improved performance. Our results suggest that a balance between the additional difficulty and control information introduced by specificity can accelerate learning.

Fetch Results We demonstrate accelerated learning in `Push`, `PickAndPlace` and `Slide` by (1) expanding the dimensionality of the `Push` and `PickAndPlace` goal spaces using feature Protoges, and (2) increasing the specificity of the `Slide` goal space using an epsilon Protoge. Test success is always measured with respect to the original goal space. See

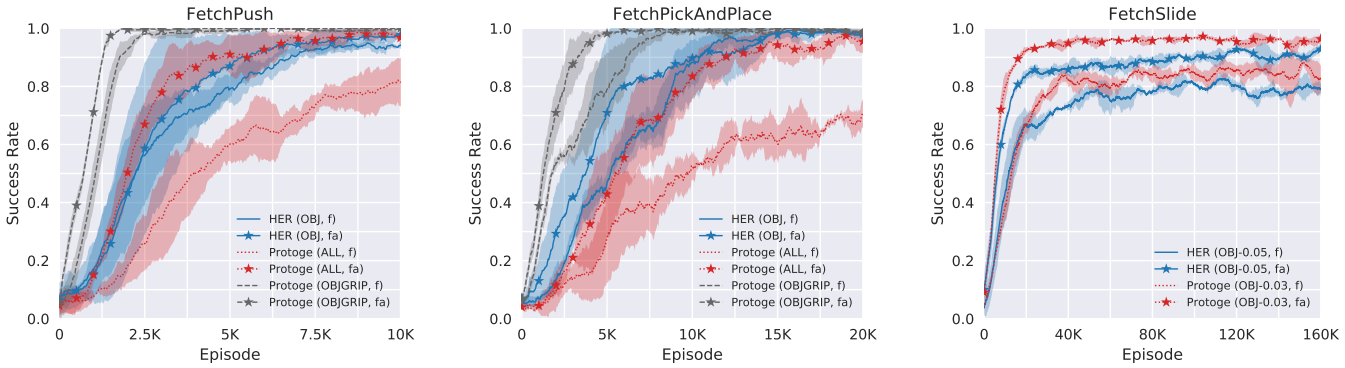


Fig. 3: In `Push` and `PickAndPlace`, the standard goal representation `OBJ` is greatly outperformed by the finer `OBJGRIP` representation, which uses 6-dimensional Protoges. The `ALL` state-goal representation does not do as well as `OBJGRIP`, but is able to a learn even when thresholds are hard-coded. In `Slide`, our epsilon Protoge (`OBJ-0.03`) approach outperforms the standard `OBJ-0.05` approach. In all cases, the `futureactual` strategy (fa) performed better than the `future` strategy (f). When used together, Protoge and `futureactual` establish a new state-of-the-art. Our results can be compared to the baselines of Plappert et al. (2018) [13] by noting that our 10K episodes are roughly equal to 5 Plappert epochs. Our greatly improved baseline results are due to implementation differences (see main text). For all environments, each configuration was run with the same 3 random seeds.

Figure 3. In `Push` and `PickAndPlace` we experiment with a 6-dimensional object-gripper goal specification (`OBJGRIP`), which reports both the box position and the gripper position, and a 25-dimensional state-goal specification (`ALL`), which reports all 25-dimensions of the state. In each case we use a hand-coded Protoge map to encode desired goals into the expanded goal space. The Protoge places the hand above the box, and encodes the other dimensions (e.g., speed and object rotation) in the state-goal case to have a value of 0. We use the same epsilon threshold as the original environments with respect to the original goal dimensions (thus, the expanded goal space is finer than the original goal space, and achieving an expanded goal necessarily achieves the original goal), and use an element-wise epsilon threshold for added dimensions, which we manually set to 0.05. In `Slide` we experiment with a tighter goal specification, which reduces the distance threshold for goal satisfaction from 0.05 to 0.03 (meters). We plan to experiment with automatically learning optimal thresholds in future work. We expect that learned thresholds will greatly improve performance, especially that of the state-goal representation (`ALL`).

A Novel HER Strategy As a result of the increase in goal specificity, the agent's state rarely satisfies the Protoge of any potential desired goal (e.g., the agent's gripper is rarely directly above the box); as such, when using the `future` strategy, very few desired goal Protoges are added to the agent's replay buffer. To combat this effect, we propose the `futureactual` strategy, which mixes goals sampled according to `future` with goals sampled from a buffer of past *actual* goals (in the agent's goal space; i.e., using Protoge). This focuses the agent's learning effort on actual desired goals while still providing the agent with enough initial reward signal to benefit from HER. In our experiments we always sample 80% of the agent's training experiments using HER, with 40% sampled according to `future` and 40% sampled according to `actual`. Although Protoge still works with `future`, we find that `futureactual` improves performance, even in absence of Protoge (Figure 3).

Implementation Highlights We use DDPG together with our own implementation of HER. Rather than distribute training across parallel workers (as done by Plappert et al. [13]), we train a single agent in 12 parallel environments. Our actors and critics use 3 layer-normalized [2] hidden layers of 512 units each. We train with a batch size of 1000 every two environment steps, and update our target network every 40 training steps using an update factor of 0.05. We apply L_2 action normalization with coefficient 0.1. Our `Push` and `PickAndPlace` agents use epsilon exploration with an epsilon of 0.3, whereas our `Slide` agent does not use any epsilon exploration. Other hyperparameters are similar to those used by Plappert et al. [13]. Our baseline `future` agents learn significantly faster than the agents of Plappert et al., as well as the more recent agents of Liu et al [8], most likely due to the different training regime.

4 Transfer Learning with Protoge

Finer goal representations are more flexible and allow for efficient transfer. The state-goal representation in particular, is *universal*: an agent trained on the state-goal space can potentially adapt to arbitrary goals, regardless of their form, so long as Protoges are available. This effectively reverses the HER assumption: instead of assuming that given state s we can find a goal g that is satisfied in s [1], we assume that given goal g we can find a state s that satisfies g. Our present work demonstrates transferability using hand-designed Protoge maps, but we are working towards an effective method for learning Protoges maps online.

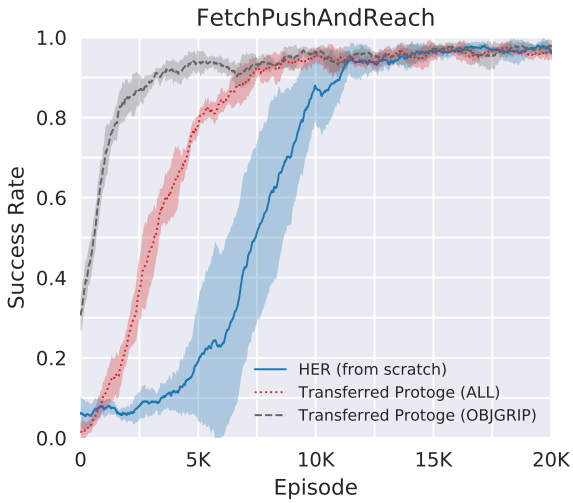


Fig. 4: Transferred Protoge agents, trained on `Push`, solve `PushAndReach` faster than an agent trained from scratch (5 random seeds each).

In Figure 4, we show that both the `OBJGRIP` and `ALL` agents from the previous Section can effectively transfer their knowledge to a `PushAndReach` environment, which has 6-dimensional goals and requires the agent to both (1) push the box to a 3-dimensional target location and (2) move its gripper to another 3-dimensional target location. The target locations are independently sampled (the box goal is sampled according to the same distribution as `Push`, and the gripper goal is sampled roughly according to the same distribution as the `FetchReach` environment). Since `PushAndReach` is harder than `Push`, we see that the standard HER approach learns slightly slower than it does in `Push`. The transferred agents are able to learn much faster, even though the distribution of desired goals has changed significantly. Note that while the transferred `OBJGRIP` agent is now using the native goal space (no Protoge), the transferred `ALL` agent is using a Protoge map, as before, to expand the native 6-dimensional goal into a 25-dimensional goal.

5 Future Work

This work is still in its early stages, and we are exploring several avenues going forward:

- In our `Fetch` experiments, we use a hard-coded element-wise success threshold for expanded goal dimensions. Can we learn these thresholds automatically? We are currently exploring a curriculum learning approach [5].
- In particular, many aspects of the state space are usually outside of the agent's control, and it is unreasonable to ask an agent to achieve arbitrary goals in the state-goal space. How can we automatically recognize controllable aspects of the state and use only those when defining Protoges? One promising approach is to learn a "controllable" latent space by predicting inverse dynamics $f(a_t|s_t, s_{t+1})$ [12].
- In our `Fetch` experiments, we use the native feature space to define goals and Protoges. How can we generalize this approach to the agent's latent feature space?
- More generally, we may wish to learn Protoge maps between two complex goal spaces (recall that two goal spaces can both be finer than the other). For example, we might want to map the natural language goal space [4], describing the task that the agent needs to perform, to raw pixels (an image) showing the agent's first person view [11], or vice versa. How can we learn such maps automatically?
- For a given coarse goal, there are many candidate Protoges, any of which satisfy the original goal. It would be interesting explore the generation of optimal Protoges, conditioned on the current state and goal.
- In our experiments, we used the original `Push` goal space to generate Protoges in order to train the `ALL` agent, which agent was able to quickly transfer its knowledge to `PushAndReach`. It seems likely, however, that the `ALL` agent could have designed its own goal curriculum, separately from `Push`, and trained itself in an unsupervised fashion by taking advantage of, e.g., curiousity [12].

References

[1] ANDRYCHOWICZ, M., WOLSKI, F., RAY, A., SCHNEIDER, J., FONG, R., WELINDER, P., MCGREW, B., TOBIN, J., ABBEEL, O. P., AND ZAREMBA, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems* (2017), pp. 5048–5058.

[2] BA, J. L., KIROS, J. R., AND HINTON, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] BROCKMAN, G., CHEUNG, V., PETTERSSON, L., SCHNEIDER, J., SCHULMAN, J., TANG, J., AND ZAREMBA, W. Openai gym, 2016.

[4] CHAN, H., WU, Y., KIROS, J., FIDLER, S., AND BA, J. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *arXiv preprint arXiv:1902.04546* (2019).

[5] EPPE, M., MAGG, S., AND WERMTER, S. Curriculum goal masking for continuous deep reinforcement learning. *arXiv preprint arXiv:1809.06146* (2018).

[6] JADERBERG, M., MNIH, V., CZARNECKI, W. M., SCHAUL, T., LEIBO, J. Z., SILVER, D., AND KAVUKCUOGLU, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* (2016).

[7] LILLICRAP, T. P., HUNT, J. J., PRITZEL, A., HEESS, N., EREZ, T., TASSA, Y., SILVER, D., AND WIERSTRA, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[8] LIU, H., TROTT, A., SOCHER, R., AND XIONG, C. Competitive experience replay. In *International Conference on Learning Representations* (2019).

[9] LOCKE, E. A., SHAW, K. N., SAARI, L. M., AND LATHAM, G. P. Goal setting and task performance: 1969–1980. *Psychological bulletin 90*, 1 (1981), 125.

[10] MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[11] NAIR, A. V., PONG, V., DALAL, M., BAHL, S., LIN, S., AND LEVINE, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems* (2018), pp. 9209–9220.

[12] PATHAK, D., AGRAWAL, P., EFROS, A. A., AND DARRELL, T. Curiosity-driven exploration by self-supervised prediction. In *ICML* (2017).

[13] PLAPPERT, M., ANDRYCHOWICZ, M., RAY, A., MCGREW, B., BAKER, B., POWELL, G., SCHNEIDER, J., TOBIN, J., CHOCIEJ, M., WELINDER, P., ET AL. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464* (2018).

[14] SCHAUL, T., HORGAN, D., GREGOR, K., AND SILVER, D. Universal value function approximators. In *International Conference on Machine Learning* (2015), pp. 1312–1320.

[15] SUTTON, R. S., AND BARTO, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[16] SUTTON, R. S., MODAYIL, J., DELP, M., DEGRIS, T., PILARSKI, P. M., WHITE, A., AND PRECUP, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (2011), pp. 761–768.

Measuring how people learn how to plan

Yash Raj Jain
Rationality Enhancement Group, MPI for Intelligent Systems, Tübingen, Germany
Birla Institute of Technology & Science-Pilani, Hyderabad, India

Frederick Callaway
Department of Psychology, Princeton University, NJ, USA

Falk Lieder
Rationality Enhancement Group, MPI for Intelligent Systems, Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany

Abstract

The human mind has an unparalleled ability to acquire complex cognitive skills, discover new strategies, and refine its ways of thinking and decision-making; these phenomena are collectively known as *cognitive plasticity*. One important manifestation of cognitive plasticity is learning to make better – more far-sighted – decisions via planning. A serious obstacle to studying how people learn how to plan is that cognitive plasticity is even more difficult to observe than cognitive strategies are. To address this problem, we develop a computational microscope for measuring cognitive plasticity and validate it on simulated and empirical data. Our approach employs a process tracing paradigm recording signatures of human planning and how they change over time. We then invert a generative model of the recorded changes to infer the underlying cognitive plasticity. Our computational microscope measures cognitive plasticity significantly more accurately than simpler approaches, and it correctly detected the effect of an external manipulation known to promote cognitive plasticity. We illustrate how computational microscopes can be used to gain new insights into the time course of metacognitive learning and to test theories of cognitive development and hypotheses about the nature of cognitive plasticity. Future work will leverage our computational microscope to reverse-engineer the learning mechanisms enabling people to acquire complex cognitive skills such as planning and problem solving.

Keywords: cognitive plasticity; planning; decision-making; process-tracing; statistical methods

1 Introduction

The way we think and decide is remarkably plastic. This *cognitive plasticity* enables people to learn how to make better decisions. Despite initial research on how people acquire cognitive skills (van Lehn, 1996), the underlying learning mechanisms are still largely unknown. Reverse-engineering how people learn how to think and how to decide is very challenging because we can neither observe people's cognitive strategies, nor how they change with experience – let alone the underlying learning mechanisms. To make these learning mechanisms visible, we develop a computational microscope for measuring how people learn how to plan. Our method has two components: i) a recently developed process-tracing paradigm that renders people's behavior highly diagnostic of their planning strategies, and ii) the inversion of a generative model of how changes in this observable behavior arise from cognitive plasticity. Our computational microscope makes it possible to observe how people's planning strategies change from each decision to the next. This sheds new light on the time course and the nature of metacognitive learning and allows us to test theories of cognitive development and characterize the effects of feedback and individual differences on cognitive plasticity.

2 Methods

2.1 Process-tracing using the Mouselab-MDP paradigm

Planning, like all cognitive processes, cannot be observed directly but has to be inferred from observable behavior. This is generally an ill-posed problem. To address this challenge, researchers have developed *process-tracing* methods that elicit behavioral signatures of latent cognitive processes. In the Mouselab paradigm, for example, decision strategies can be traced by recording the order in which people inspect the payoffs of different gambles (Payne, Bettman, & Johnson, 1993). While these behavioral signatures are still indirect measures of cognitive processes, they do provide additional information about what the underlying cognitive strategy might be.

Here, we employ an extension of the Mouselab paradigm to the domain of sequential decision making (Callaway, Lieder, Krueger, & Griffiths, 2017; Callaway et al., 2018). In the Mouselab-MDP paradigm, illustrated in Figure 1a, participants are presented with a series of route planning problems in which each location (the gray circles) harbors a gain or loss. On each trial, participants choose among the six possible paths to maximize the total reward they receive across the three locations on the path. Initially, these rewards are occluded; however, participants can reveal the reward at a each location by clicking on it. This explicit clicking action corresponds to evaluating the quality of a future state, a fundamental cognitive operation in planning. The cognitive cost of this operation is externalized by an explicit cost of one point for each reward revealed. Participants are thus encouraged to not immediately click every location, but instead reveal information as necessary. The order of clicks reveals, for example, whether participants engage in forwards planning (starting from the current state), or backwards planning (starting from the possible end states).

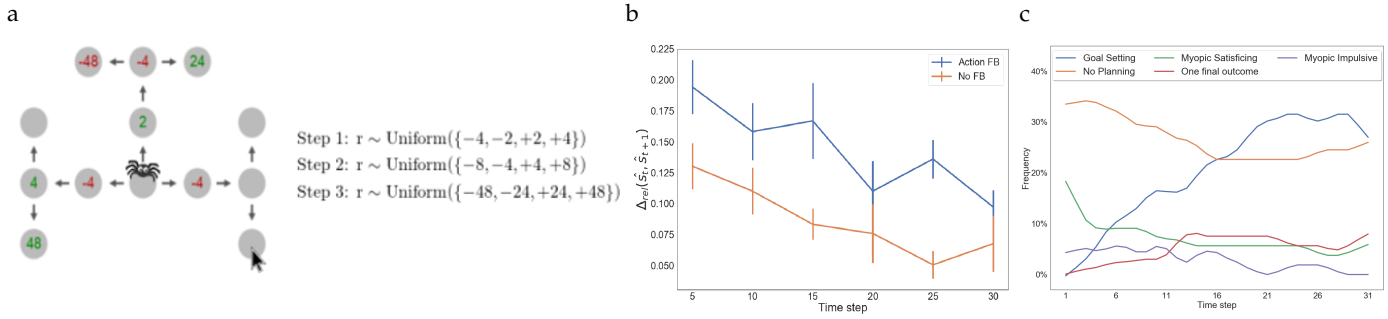


Figure 1: a) Illustration of the Mouselab-MDP paradigm. Rewards are revealed by clicking. b) Feedback boosted cognitive plasticity at the beginning of learning. c) Time course of the frequencies of the five most common strategies.

2.2 Measurement model

To construct a computational microscope for measuring cognitive plasticity, we model the trial-by-trial sequence of peoples' cognitive strategies $(S_1, S_2, \cdots, S_{31})$ as a hidden Markov chain that emits the observed process tracing data. Our models require methodological assumptions about i) how cognitive strategies manifest in process-tracing data, ii) the space of cognitive mechanisms that can be learned, and iii) the nature and amount of cognitive plasticity that might occur. The following paragraphs detail our assumptions about each of these three components in turn.

Observation model. Our observation model thus specifies the probability of observing a sequence of clicks $\mathbf{d_t}$ on trial t if the strategy was S_t (i.e., $P(\mathbf{d_t}|S_t)$). To achieve this, we quantify each strategy's propensity to generate a click c (or stop collecting information) in belief state b by a weighted sum of the features $f_1(b,c),\cdots,f_{29}(b,c)$. The belief state encodes observed rewards.The features describe the click c relative to this information (e.g., by the value of the largest reward that can be collected from the inspected location) and in terms of the action it gathers information about (e.g., whether it pertains to the first, second, or third step)[1]. The features and weights jointly determine the strategy's propensity to make click c in belief state b:

$$P(\mathbf{d_t}|S_t) = \prod_{i=1}^{|\mathbf{d_t}|} \frac{\exp\left(\frac{1}{\tau}\cdot\sum_{k=1}^{|w^{(S)}|} w_k^{(S)}\cdot f_k^{(S)}(c_{t,i},b_{t,i})\right)}{\sum_{c\in\mathcal{C}_{b_t}}\exp\left(\frac{1}{\tau}\cdot\sum_{k=1}^{|w^{(S)}|} w_k^{(S)}\cdot f_k^{(S)}(c,b_{t,i})\right)}, \tag{1}$$

where $d_{t,i}$ is the i$^{\text{th}}$ click in trial t (or the decision to stop clicking), τ is the decision temperature, and $w^{(S)}$ are the weights of strategy S.

Space of cognitive mechanisms. The process tracing data from Lieder (2018) suggested that people use 38 different planning strategies ($\mathcal{S}$)[1]). These strategies differ in how much they plan (ranging from none to all), which information they focus on, and in which order they collect it. Building on the observation model in Equation 1, we represent each strategy by a weight vector $\mathbf{w} = (w_1,\cdots,w_{29})$ that specifies its preference for more vs. less planning, considering immediate vs. long-term consequences, satisficing vs. maximizing, avoiding losses, and other desiderata.

Ward's hierarchical clustering method suggested 11 types of planning strategies including acting impulsively without any planning, five types of goal-setting strategies, three types of forward-planning strategies similar to depth-first search, best-first search, and breadth-first search respectively. To use this method we defined the distance $\Delta(s_1,s_2)$ between strategy s_1 and s_2 as the Jensen-Shannon divergence (Lin, 1991) between the distributions of click sequences and belief states induced by strategies s_1 and $s_2$and approximated it using Monte-Carlo integration.

Prior on strategy sequences. Inferring a strategy from a single click sequence could be unreliable. Our method therefore exploits temporal dependencies between subsequent strategies to smooth out its inferences. Transitions from one strategy to the next can be grouped into three types: repetitions, gradual changes, and abrupt changes. While most neuroscientific and reinforcement-learning perspectives emphasize gradual learning, others suggest that animals change their strategy abruptly when they detect a change in the environment (Gershman, Blei, & Niv, 2010). Symbolic models and stage theories of cognitive development also assume abrupt changes (e.g., Piaget, 1971), and it seems plausible that both types of mechanisms might coexist. To accommodate these perspectives, we consider three prior distributions on participants' trial-by-trial sequence of cognitive strategies.

The *gradual learning prior* in Equation 2 assumes that strategies changes gradually, that is

$$P(S_{t+1}=s|S_t,m_{\text{gradual}}) = \frac{\exp(-\frac{1}{\tau}\cdot\Delta(s,S_t))}{\sum_{s'\in\mathcal{S}}\exp(-\frac{1}{\tau}\cdot\Delta(s',S_t))}, \tag{2}$$

where $|\mathcal{S}|$ is the number of strategies. The *abrupt changes prior* assumes that transitions are either repetitions or jumps:

$$P(S_{t+1}=s|S_t,m_{\text{abrupt}}) = p_{\text{stay}}\cdot\mathbb{I}(S_{t+1}=S_t) + (1-p_{\text{stay}})\cdot\frac{\mathbb{I}(s\neq S_t)}{|\mathcal{S}|-1}. \tag{3}$$

Finally, the *mixed prior* in Equation 4 assumes that both types of changes coexist.

$$P(S_{t+1}=s|S_t,m_{\text{mixed}}) = p_{\text{gradual}}\cdot P(S_{t+1}=s|S_t,m_{\text{gradual}}) + (1-p_{\text{gradual}})\cdot P(S_{t+1}=s|S_t,m_{\text{abrupt}}). \tag{4}$$

In either case, we model the probability of the first strategy by a uniform distribution over the space of decision strategies.

Together with the observation model and the strategy space described above each of these priors defines a generative model of a participant's process tracing data $\mathbf{d}$; this model has the following form:

$$P(\mathbf{d},S_1,\cdots,S_T) = \frac{1}{|\mathcal{S}|}\cdot\prod_{t=2}^{T} P(S_t|S_{t-1},m)\cdot P(\mathbf{d_t}|S_t). \tag{5}$$

The three measurement models differ in the identity of $m\in\{m_{\text{gradual}},m_{\text{abrupt}},m_{\text{mixed}}\}$. Inverting these models gives rise to a computational method for measuring cognitive plasticity.

[1]The features were devised to be able to capture the whole range of strategies exhibited by our participants. A detailed description of the features and strategies is available at `https://osf.io/y58d3/?view_only=fa2f89de3aa04d4d87af3d050bb1a64c`

2.3 Computational microscopy by model inversion

The models above describe how cognitive plasticity manifests in process-tracing data. To measure cognitive plasticity we have to reason backwards from the process tracing data to cognitive changes that generated it. That is, we can build a computational microscope for measuring cognitive plasticity by inverting these measurement models. To achieve this, we leverage the Viterbi algorithm to compute maximum a posteriori (MAP) estimates of the hidden sequence of planning strategies given the observed process tracing data, the measurement model, and its parameters (p_{stay} for m_{abrupt} and p_{gradual} and p_{stay} for m_{mixed}).

3 Validating the computational microscope

Validation on synthetic data. To validate our computational microscope, we apply it to simulated process tracing data. To avoid bias towards any one measurement model, we sampled 100 simulated trials from each of the three measurement models and combined them into a single data set. We then invert the three measurement models on each of the simulated trials ($\mathbf{d}$) and compared the MAP estimate of each strategy sequence ($\hat{\mathbf{S}}$) against the ground truth (S) in terms of the proportion of correctly inferred strategies and the distance between the inferred strategies and the ground truth.

As a baseline, we evaluated the computational method that inverts the observation model in Equation 1 on each click sequence independently. This simple approach was sufficient to infer the correct strategy about 81% of the time (95% confidence interval: $[80.2\%, 81.8\%]$). The average distance Δ from the inferred strategy to the true one was only 21% of the average distance from each strategy to its closest neighbor ($\Delta_{\text{rel}}(\hat{\mathbf{s}}^{\text{baseline}}, \mathbf{s}) = 0.215$, 95% confidence interval: $[0.20, 0.23]$). This shows that the simulated click sequences were highly diagnostic of the strategies that generated them.

We found that exploiting temporal dependencies among subsequent strategies by using our measurement models significantly improved the proportion of correctly inferred strategies to 88.5%, 88.3%, and 88.5% for m_{gradual}, m_{abrupt}, and m_{mixed} respectively (all $p < 0.0001$) and decreased the average distance between the inferred strategies and the ground truth by more than 40% ($\Delta_{\text{rel}}(\hat{\mathbf{s}}^{\text{gradual}}, \mathbf{s}) = 0.124$, $\Delta_{\text{rel}}(\hat{\mathbf{s}}^{\text{mixed}}, \mathbf{s}) = 0.124$, and $\Delta_{\text{rel}}(\hat{\mathbf{s}}^{\text{abrupt}}, \mathbf{s}) = 0.127$, all $p < 0.0001$). These results suggest that – under reasonable, theory-agnostic assumptions about what cognitive plasticity might be like – our computational microscope is more accurate than simpler methods.

Validation on empirical data. To validate our computational microscope on empirical data, we applied it to the Mouselab-MDP process-tracing data from Experiments 1–3 by Lieder (2018) where 176 participants solved 31 different 3-step planning problems of the form shown in Figure 1a. We asked if our computational microscope can detect the effect of the feedback participants in the second condition of Experiment 1 received on the (sub)optimality of their chosen actions. Our computational microscope successfully detected this manipulation. As shown in Figure 1b, the inferred learning-induced changes were significantly larger in the feedback condition than in the control condition in the first 15 trials and in trials 21–25 (all $p \leq 0.012$). Action feedback selectively increased the probability of 8 performance-increasing strategy changes (and only 2 performance-decreasing ones) while decreasing the probability of 5 performance-decreasing transitions, 5 self-transitions, and only 1 performance-increasing transition. Our method's ability to detect the plasticity-enhancing effects of feedback suggests that its inferences provide a valid measure of cognitive plasticity. Figure 1b also shows that cognitive plasticity slowed down as participants adapted to the experiment's stationary environment.

4 Shedding light on cognitive plasticity

Having validated our computational microscope on both simulated and empirical data, we now apply it to measure how people learned how to plan in the control condition of Experiment 1 and the training phase of the control conditions of Experiments 2 and 3 from Lieder (2018).

Temporal evolution of strategy frequencies. As shown in Figure 1c, we found that the most common initial strategy was to act impulsively without any planning (*No Planning*). Over time the prevalence of this strategy decreased from 34% to 26% ($\chi^2(1) = 7.95$, $p = 0.0048$).Conversely, the frequency of the near-optimal *Goal Setting* strategy increased from 4% to 30% ($\chi^2(1) = 148.85, p < .0001$). The frequencies of the two maladaptive strategies that decide based on immediate rewards (*Myopic Satisficing* and *Myopic Impulsive*) dropped from about 11% to 5% ($\chi^2(1) = 11.74, p = .0006$) and from and 4% to 0.6% ($\chi^2(1) = 11.62, p = 0.0006$) respectively, whereas the frequency of the strategy *One Final Outcome* that prioritizes long-term consequences increased from 1% to 6% ($\chi^2(1) = 20.22, p < 0.0001$).

Testing theories of cognitive development. Prominent theories disagree about whether cognitive development is a gradual process (Siegler, 1996) or proceeds in discrete stages (Piaget, 1971) with abrupt transitions. Our computational microscope suggested that cognitive plasticity includes both gradual and abrupt strategy changes. The majority of inferred strategy changes was gradual (i.e., 59.1%, $\chi^2(1) = 56.8$, $p < 0.0001$) but there was also a non-negligible percentage

of abrupt changes (i.e., 40.9%). Consistent with Siegler's overlapping waves theory (Siegler, 1996) we found that i) multiple different strategies were being used at each point in time throughout the learning process (i.e., 2.2 strategies on average), and ii) high-performing strategies become more prevalent over time whereas low-performing strategies become less prevalent (i.e., there was a significant rank correlation between each strategy's average performance and the change in its frequency; Spearman's $\rho(37) = 0.39$, $p = 0.0154$).

Learning trajectories. To identify the most common learning trajectories, we categorized each inferred strategy as belonging to one of the 11 types of strategies described earlier and extracted the order in which they appeared. We found that 86.0% of the learning trajectories were unique and the remaining trajectories were exhibited by only 2–4 learners each. Zooming in on the 49 participants who learned the near optimal goal setting strategy, we found that they reached it via 38 unique learning trajectories. Consistent with the overlapping waves theory, we found that 83.8% of these learning trajectories included at least one intermediary strategy and identified three gateways to optimal planning: 35% of the intermediary strategies inspected all potential final states – whereas the optimal strategy stops once it encounters the best possible outcome – and sometimes planned backwards from undesirable states; 27% inspected potential final states like the optimal strategy but wastefully inspected paths towards undesirable final outcomes, and 21% of the penultimate strategies inspected both immediate and final outcomes while ignoring the intermediate states. Interestingly, all of these intermediary strategies performed gratuitous planning operations.

5 Discussion

We have successfully validated our method on both synthetic and human data. The results suggest that our computational microscope can measure cognitive plasticity in terms of the temporal evolution of people's cognitive strategies. We believe this method has great potential for uncovering the mechanisms of cognitive plasticity and how they are impacted by the learning environment, individual differences, time pressure, motivation, and interventions – including feedback and instructions. We are optimistic that computational microscopes will become useful tools for reverse-engineering the learning mechanisms that enable people to acquire complex cognitive skills and shape how we think and decide. To make this possible, we will extend the proposed measurement model to continuous strategy spaces defined in terms of the interaction between the goal-directed system, the Pavlovian system, and habits (O'Doherty, Cockburn, & Pauli, 2017; van der Meer, Kurth-Nelson, & Redish, 2012). Our findings should be taken with a grain of salt because a more psychologically plausible distance metric or a more realistic strategy representation could lead to different conclusions. Future work will evaluate our method on synthetic data generated according to extant models of cognitive plasticity and quantify how well the inferred strategy sequences explain empirical data. Our approach makes it possible to more directly observe the previously hidden phenomenon of cognitive plasticity in all of its facets – ranging from skill acquisition and cognitive development to the progression of psychiatric symptoms and mental disorders. Finally, reverse-engineering people's ability to discover and continuously refine their own algorithms could enable substantial advances towards self-improving (general) artificial intelligence.

References

Callaway, F., Lieder, F., Das, P., Gul, S., Krueger, P. M., & Griffiths, T. L. (2018). A resource-rational analysis of human planning. In *Proceedings of the 40th annual conference of the cognitive science society.*

Callaway, F., Lieder, F., Krueger, P. M., & Griffiths, T. L. (2017). Mouselab-mdp: A new paradigm for tracing how people plan. In *The 3rd multidisciplinary conference on reinforcement learning and decision making.*

Gershman, S. J., Blei, D. M., & Niv, Y. (2010). Context, learning, and extinction. *Psychological review*, *117*(1), 197.

Lieder, F. (2018). Developing an intelligent system that teaches people optimal cognitive strategies. In F. Lieder (Ed.), *Beyond bounded rationality: Reverse-engineering and enhancing human intelligence* (chap. 8).

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory*, *37*(1), 145–151.

O'Doherty, J. P., Cockburn, J., & Pauli, W. M. (2017). Learning, reward, and decision making. *Annual review of psychology*, *68*, 73–100.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1993). *The adaptive decision maker*. Cambridge university press.

Piaget, J. (1971). *The theory of stages in cognitive development.* McGraw-Hill.

Siegler, R. S. (1996). *Emerging minds: The process of change in children's thinking*. New York: Oxford University Press.

van der Meer, M., Kurth-Nelson, Z., & Redish, A. D. (2012). Information processing in decision-making systems. *The Neuroscientist*, *18*(4), 342–359.

van Lehn, K. (1996). Cognitive skill acquisition. *Annual review of psychology*, *47*(1), 513–539.

A continuity result for optimal memoryless planning in POMDPs

Johannes Rauh
MPI MIS
jarauh@mis.mpg.de

Nihat Ay
MPI MIS / Uni Leipzig / Santa Fe Institute
nay@mis.mpg.de

Guido Montúfar
UCLA Math and Stat / MPI MIS
montufar@math.ucla.edu

Abstract

Consider an infinite horizon partially observable Markov decision process. We show that the optimal discounted reward under memoryless stochastic policies is continuous under perturbations of the observation channel. This implies that we can find approximately optimal memoryless policies by solving an approximate problem with a simpler observation channel.

Keywords: POMDPs, memoryless stochastic policy, optimal policy

1 Introduction

Policy optimization in partially observable Markov decision processes (POMDPs) is known to be a difficult problem. In order to better understand this problem, we can study special cases where the system has some additional structure (e.g., the observations identify the world state to within a few possibilities), or we can also restrict the optimization problem to policies with some additional structure (e.g., memoryless policies). The optimization problem over memoryless policies has been discussed in various works (see, e.g., Ross, 1983; Vlassis et al., 2012; Azizzadenesheli et al., 2016).

In this context, the connections between information, memory and value are of particular interest (see Kaelbling et al., 1998). We are interested in the relations that exist between the observation channel, on the one hand, and the structure of the optimal memoryless policies, on the other hand. In particular, we are interested in whether certain types of POMDP optimization problems allow for optimal or nearly optimal policies that have a particularly simple structure.

Previous work in this direction has characterized families of policies that contain optimal memoryless policies for any POMDP of a particular type (see Montúfar and Rauh, 2017; Montúfar et al., 2015; Montúfar et al., 2015). In particular, these works consider the number of actions that a memoryless policy needs to randomize at a given observation, depending on the number of world states that are compatible with that observation. In other words, depending on the properties of the observation channel, they conclude that there exists a simple optimal memoryless policy. A natural question is: If the observation channel nearly satisfies the conditions under which it is known that a simple optimal policy exists, can we conclude that there exists a simple policy that is nearly optimal? In this short article, we show that this is indeed the case. Thereby, we contribute to the understanding of approximate memoryless stochastic planning in POMDPs.

2 POMDPs and localization of optimal policies

We briefly introduce the definitions and settings.

Definition 1. A *Partially Observed Markov Decision Process* (POMDP) is a tuple $(\mathcal{W}, \mathcal{S}, \mathcal{A}, \alpha, \beta, R)$ consisting of

1. finite sets $\mathcal{W}$ (world states), $\mathcal{S}$ (sensor states/observations) and $\mathcal{A}$ (actions),
2. Markov kernels/channels $\alpha : \mathcal{W} \times \mathcal{A} \to \mathcal{W}$ (world state transition) and $\beta : \mathcal{W} \to \mathcal{S}$ (observation channel),
3. and a reward function $R : \mathcal{W} \times \mathcal{A} \to \mathbb{R}$.

A Markov decision process (MDP) is the special case where β conveys full information about the world state.

We consider time independent memoryless stochastic policies.

Definition 2. A *policy* is a Markov kernel $\pi : \mathcal{S} \to \mathcal{A}$. Denote $\Delta_{S,A}$ the set of all such policies.

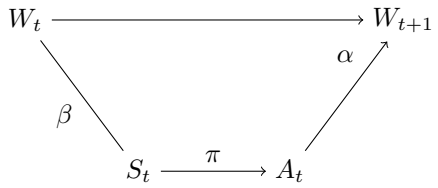


Figure 1: The graphical structure of a POMDP.

A POMDP, a policy π, and a starting distribution μ on $\mathcal{W}$, together, define a stochastic process $(W^t, S^t, A^t)_t$, which is a sequence of random variables $(W^t)_t$ (world states), $(S^t)_t$ (sensor states), and $(A^t)_t$ (actions). The graphical structure is shown in Figure 1.

We consider infinite horizon discounted rewards.

Definition 3. Given a POMDP and a *policy* π, the *discounted reward* with discount factor γ is

$$\mathcal{R}_\gamma(\pi) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \sum_{w,a} R(w,a)P(W^t = w, A^t = a).$$

We denote $\mathcal{R}^*_\gamma = \sup_{\pi\in\Delta_{S,A}} \mathcal{R}_\gamma(\pi)$ the optimal value over the set of memoryless stochastic policies.

We are interested in the following theorem by Montúfar and Rauh (2017), which provides a type of extension of the well known fact that any MDP has an optimal policy over the set of memoryless stochastic policies which is deterministic. Given $\pi\colon \mathcal{S} \to \mathcal{A}$, let $\operatorname{supp}(\pi(\cdot|s)) = \{a \in \mathcal{A}\colon \pi(a|s) > 0\}$.

Theorem 4. *Consider a POMDP $(W, S, A, \alpha, \beta, R)$. Then there is a policy $\pi^* \in \Delta_{S,A}$ with $|\operatorname{supp}(\pi^*(\cdot|s))| \leq |\operatorname{supp}(\beta(s|\cdot))|$ for all $s \in S$, and $\mathcal{R}_\gamma(\pi^*) \geq \mathcal{R}_\gamma(\pi)$ for all $\pi \in \Delta_{S,A}$.*

This result implies that, in order to optimize a POMDP over the set of memoryless stochastic policies, it suffices to consider a subset of policies. This can be translated into a choice of a policy model with few parameters or also into heuristics for the policy optimization. The result is optimal in the sense that there are POMDPs $(W, S, A, \alpha, \beta, R)$ where each policy $\pi^* \in \Delta_{S,A}$ with $\mathcal{R}_\gamma(\pi^*) \geq \mathcal{R}_\gamma(\pi)$ for all $\pi \in \Delta_{S,A}$ satisfies $|\operatorname{supp}(\pi^*(\cdot|s))| \geq |\operatorname{supp}(\beta(s|\cdot))|$.

A problem with Theorem 4 is that, in concrete situations, we might not be able to exclude world states with absolute certainty, meaning that $|\operatorname{supp}(\beta(s|\cdot))| = |\mathcal{W}|$. Moreover, the number of world states might be as large or larger than the number of possible actions, $|\mathcal{W}| \geq |\mathcal{A}|$, in which case the statement of the theorem is vacuous.

3 Continuity of the reward and localization of nearly optimal policies

In order to remedy the shortcomings of Theorem 4, we need a continuous version of the characterization. Our continuous extension is as follows. We show that if the observation channel β is close to some other channel β', in an appropriate sense, then we can find a near to optimal policy π with $|\operatorname{supp}(\pi(\cdot|s))| \leq |\operatorname{supp}(\beta'(s|\cdot))|$ for all $s \in S$.

Theorem 5. *Consider a POMDP with $\gamma < 1$ and $\|R\|_\infty := \max_{w,a} |R(w,a)|$. Let β' be a Markov kernel $\mathcal{W} \to \mathcal{S}$ that satisfies*

$$\|\beta(\cdot|w) - \beta'(\cdot|w)\|_{TV} = \frac{1}{2}\sum_s \big|\beta(s|w) - \beta'(s|w)\big| \leq \epsilon \quad \text{for all } w \in \mathcal{W}.$$

Then there is a policy π that satisfies $|\operatorname{supp}(\pi(\cdot|s))| \leq |\operatorname{supp}(\beta'(s|\cdot))|$ for all s and $\mathcal{R}_\gamma(\pi) \geq \mathcal{R}^*_\gamma - 2\frac{\epsilon}{1-\gamma}\|R\|_\infty$.

We prove this theorem based on a continuity result for the discounted reward function with respect to the observation channel.

Theorem 6. *Consider two POMDPs $(\mathcal{W}, \mathcal{S}, \mathcal{A}, \alpha, \beta, R)$ and $(\mathcal{W}, \mathcal{S}, \mathcal{A}, \alpha, \beta', R)$ that satisfy*

$$\|\beta(\cdot|w) - \beta'(\cdot|w)\|_{TV} = \frac{1}{2}\sum_s \big|\beta(s|w) - \beta'(s|w)\big| \leq \epsilon \quad \text{for all } w \in \mathcal{W}.$$

Then the discounted reward functions $\mathcal{R}_\gamma$, $\mathcal{R}'_\gamma$ of the two POMDPs satisfy

$$|\mathcal{R}_\gamma(\pi) - \mathcal{R}'_\gamma(\pi)| \leq \frac{\epsilon}{1-\gamma}\|R\|_\infty \quad \text{for any policy } \pi \in \Delta_{S,A} \text{ and any } \gamma < 1,$$

where $\|R\|_\infty := \max_{w,a} |R(w,a)|$. This implies, in particular, $|\mathcal{R}^*_\gamma - \mathcal{R}'^*_\gamma| \leq \frac{\epsilon}{1-\gamma}\|R\|_\infty$.

We present the proofs in the following section.

4 Proofs of the continuity result

Lemma 7. *Under the assumptions of Theorem 6, denote by A^t, W^t the action and world process of the POMDP with β, and denote by A'^t, W'^t the action and world process of the POMDP with β', where both POMDPs are controlled by the same policy π. Then,*

$$|\Pr(A^t = a|W^t = w) - \Pr(A'^t = a|W'^t = w)| \le \epsilon \qquad \text{for all } a, w.$$

Proof. (Proof of Lemma 7) The inequality follows from

$$\begin{aligned}
&\Big|\Pr(A^t = a|W^t = w) - \Pr(A'^t = a|W'^t = w)\Big| \\
&\quad = \Big|\sum_s \pi(a|s)(\beta(s|w) - \beta'(s|w))\Big| \\
&\quad = \Big|\sum_{s:\beta(s|w)\ge\beta'(s|w)} \pi(a|s)(\beta(s|w) - \beta'(s|w)) - \sum_{s:\beta'(s|w)>\beta(s|w)} \pi(a|s)(\beta'(s|w) - \beta(s|w))\Big| \\
&\quad \le \max\left\{\sum_{s:\beta(s|w)\ge\beta'(s|w)} \pi(a|s)(\beta(s|w) - \beta'(s|w)), \sum_{s:\beta'(s|w)>\beta(s|w)} \pi(a|s)(\beta'(s|w) - \beta(s|w))\right\} \\
&\quad \le \max\left\{\sum_{s:\beta(s|w)\ge\beta'(s|w)} (\beta(s|w) - \beta'(s|w)), \sum_{s:\beta'(s|w)>\beta(s|w)} (\beta'(s|w) - \beta(s|w))\right\} \\
&\quad = \|\beta'(\cdot|w) - \beta(\cdot|w)\|_{\mathrm{TV}}.
\end{aligned}$$

□

Lemma 8. *Under the assumptions of Lemma 7, for all $t \ge 0$,*

$$\sum_{aw} |\Pr(A^t W^t = aw) - \Pr(A'^t W'^t = aw)| \le (t+1)\epsilon, \tag{1a}$$

$$\sum_{w} |\Pr(W^t = w) - \Pr(W'^t = w)| \le t\epsilon. \tag{1b}$$

Proof. (Proof of Lemma 8) The proof is by induction. For $t = 0$, $\Pr(W^0 = w) = \Pr(W'^0 = w)$, so (1b) holds for $t = 0$. Assuming that (1b) holds for some t,

$$\begin{aligned}
&\sum_{aw} |\Pr(A^t W^t = aw) - \Pr(A'^t W'^t = aw)| \\
&\qquad \le \sum_w |\Pr(W^t = w) - \Pr(W'^t = w)| \sum_a |\Pr(A^t = a|W^t = w)| \\
&\qquad\quad + \sum_{aw} |\Pr(A^t = a|W^t = w) - \Pr(A'^t = a|W'^t = w)||\Pr(W'^t = w)| \\
&\qquad \le \sum_w |\Pr(W^t = w) - \Pr(W'^t = w)| \\
&\qquad\quad + \sup_w \sum_a |\Pr(A^t = a|W^t = w) - \Pr(A'^t = a|W'^t = w)| \\
&\qquad \le t\epsilon + \epsilon = (t+1)\epsilon.
\end{aligned}$$

Assuming that (1a) holds for $t-1$,

$$\begin{aligned}
&\sum_w |\Pr(W^t = w) - \Pr(W'^t = w)| \\
&\qquad = \sum_w \Big|\sum_{a,w'} \alpha(w|a,w')\big(\Pr(A^{t-1}W^{t-1} = aw') - \Pr(A'^{t-1}W'^{t-1} = aw')\big)\Big| \\
&\qquad \le \sum_{a,w'}\sum_w \alpha(w|a,w')\Big|\Pr(A^{t-1}W^{t-1} = aw') - \Pr(A'^{t-1}W'^{t-1} = aw')\Big| \\
&\qquad \le \sum_{a,w'} \Big|\Pr(A^{t-1}W^{t-1} = aw') - \Pr(A'^{t-1}W'^{t-1} = aw')\Big| \le t\epsilon.
\end{aligned}$$

□

Proof. (Proof of Theorem 6) Using

$$\sum_{t=0}^{\infty}(t+1)\gamma^t = \frac{1}{\gamma}\sum_{t=1}^{\infty} t\gamma^t = \frac{\partial}{\partial\gamma}\sum_{t=0}^{\infty}\gamma^t = \frac{\partial}{\partial\gamma}\frac{1}{1-\gamma} = \frac{1}{(1-\gamma)^2}$$

and Lemma 8,

$$\begin{aligned}
&|\mathcal{R}(\pi) - \mathcal{R}(\pi')| \\
&\quad\leq (1-\gamma)\sum_{t=0}^{\infty}\sum_{w,a}\gamma^t R(w,a)|P(W^t, A^t = w, a) - P(W'^t, A'^t = w, a)| \\
&\quad\leq (1-\gamma)\frac{1}{\gamma}\sum_{t=1}^{\infty}\gamma^t\|R\|_\infty t\epsilon = \frac{\epsilon}{1-\gamma}\|R\|_\infty.
\end{aligned}$$

□

Let $\mathcal{R}(\beta,\pi)$ be the expected reward for observation kernel β and policy π.

Lemma 9. *Let $\mathcal{R}_\gamma$, $\mathcal{R}'_\gamma$ be the discounted reward functions of two POMDPs $(\mathcal{W},\mathcal{S},\mathcal{A},\alpha,\beta,R)$, $(\mathcal{W},\mathcal{S},\mathcal{A},\alpha,\beta',R)$, and suppose that there exists $c>0$ with $|\mathcal{R}_\gamma(\pi)-\mathcal{R}'_\gamma(\pi)|\leq c$ for all policies π. If π'^* is the optimal policy for $\mathcal{R}'_\gamma$, then $\mathcal{R}^*_\gamma \geq \mathcal{R}_\gamma(\pi'^*) \geq \mathcal{R}^*_\gamma - 2c$.*

Proof. (Proof of Lemma 9) The first inequality is by definition of $\mathcal{R}^*_\gamma$. For any policy π for $\mathcal{R}_\gamma$,

$$\mathcal{R}_\gamma(\pi'^*) \geq \mathcal{R}'(\pi'^*) - c \geq \mathcal{R}'(\pi) - c \geq \mathcal{R}_\gamma(\pi) - 2c.$$

The second inequality follows when π is an optimal policy for $\mathcal{R}_\gamma$. □

Proof. (Proof of Theorem 5) This follows from Theorem 6 and Lemma 9 and Theorem 4. □

5 Discussion

We presented a continuity result that extends the applicability of previous theoretical results on the structure of optimal policies of POMDPs, and allows us to discuss approximately optimal policies. Continuity, in the way that we studied here, could be investigated not only in terms of the observation channel, but also in terms the state transition kernel. These continuity results might also serve to make statements about consistency in policy optimization in reinforcement learning, when the agent needs to estimate the world model (i.e., the kernels β and α).

Acknowledgment This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement no 757983).

References

K. Azizzadenesheli, A. Lazaric, and A. Anandkumar. Open problem: Approximate planning of pomdps in the class of memoryless policies. In V. Feldman, A. Rakhlin, and O. Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1639–1642. PMLR, 2016.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

G. Montúfar and J. Rauh. Geometry of policy improvement. In *Geometric Science of Information, LNCS 10589*, pages 282–290. Springer, 2017.

G. Montúfar, K. Ghazi-Zahedi, and N. Ay. A theory of cheap control in embodied systems. *PLoS Computational Biology*, 11(9):1–22, 2015.

G. Montúfar, K. Ghazi-Zahedi, and N. Ay. Geometry and determinism of optimal stationary control in POMDPs. *arXiv:1503.07206*, 2015.

S. M. Ross. *Introduction to Stochastic Dynamic Programming: Probability and Mathematical*. Probability and Mathematical Statistics: A Series of Monographs and Textbooks. Academic Press, Inc., 1983.

N. Vlassis, M. L. Littman, and D. Barber. On the computational complexity of stochastic controller optimization in POMDPs. *ACM Transactions on Computation Theory*, 4(4):12:1–12:8, 2012.

Posterior Sampling Networks

Vikranth R Dwaracherla
Stanford University
vikranth@stanford.edu

Benjamin Van Roy
Stanford University
bvr@stanford.edu

Morteza Ibrahimi
morteza@ibrahimi.org

Abstract

In this article, we propose a new approach for efficiently generating approximate samples from a posterior over complex models such as neural networks, induced by a prior distribution over the model family and a set of input-output data pairs. While there are other applications, we are particularly motivated in this work by its application in Thompson sampling, a technique for efficient exploration in reinforcement learning. Thompson sampling requires sampling from a posterior distribution over models, which can be achieved in special cases under restrictive assumptions. Approximations are called for when this can not be done exactly. Ensemble sampling offers an approach that is viable in complex settings such as deep reinforcement learning. However, ensemble sampling requires fitting a substantial number of separate models, which although tractable is far more computationally demanding than one would hope. We propose a new approach that is based on point estimation in an *elevated model space*. This elevated model space is made up of models that map the input space and a d-dimensional Euclidean index space to the output space. After learning the mapping, by sampling a random index, one effectively samples a random neural network that maps predictors to output. Our approach aims to learn a mapping so that this random model is approximately distributed according to the posterior over neural networks conditioned on observed data. As a sanity check, we prove that in the special case of linear models with Gaussian noise our approach can generate exact samples from the posterior. We also demonstrate empirically the efficacy of our approach in the context of bandit learning with linear and neural network models.

Keywords: approximate posterior, posterior sampling over networks

1 Introduction

It is often useful to learn not just a point estimate but a posterior distribution over models. When learning a linear model with a Gaussian prior over coefficients from observations perturbed by Gaussian noise, this can be accomplished via the Kalman filter. However, computing or even approximating a posterior distribution over more complex model classes, like neural networks, poses a far greater challenge.

Though addressing this challenge could serve many purposes, our immediate motivation is in the use of posterior samples to guide exploration in reinforcement learning. For example, approaches motivated by Thompson sampling, such as those discussed in [1], call for approximately sampling neural networks from posterior distributions. Among such approximation methods, ensemble sampling [2] stands out as one that is relatively tractable and applicable in complex problems such as deep reinforcement learning [3].

We will propose a new approach that learns a mapping from predictors and a random index to an output. After learning the mapping, by sampling a random index, one effectively samples a random model that maps predictors to output. Our approach aims to learn a mapping so that this random model is approximately distributed according to the posterior over models conditioned on observed data.

Ensemble sampling can be viewed as a special case of this approach. In ensemble sampling, K separate models are learned. The distribution of this ensemble can be viewed as an approximation to the posterior over models, in a spirit similar to particle filtering. To produce a random model with distribution approximating the posterior, a model is sampled uniformly from the ensemble. By viewing the selection in terms of sampling a random index uniformly from $\{1, \ldots, K\}$, the ensemble can be seen as a mapping from input and index to output.

Our new approach aims to offer efficiency gains over ensemble sampling by alleviating the need to learn K separate complex models. We instead learn a single model. We expect this model to be more complex than each element of the ensemble since it takes a random index as additional input. However, rather than increases the number of free parameters by a factor of K, as is done in ensemble sampling, our hope is that our new approach entails a much smaller multiple.

Our approach may sound related to generative networks [4], though there is some difference. In adversarial generative networks, a random index is mapped to an output in a manner that approximates an output distribution reflected by observed data. [5] presents some crucial shortcomings of some existing generative methods like [6], [7] for posterior approximation through some sanity checks. Please refer to [5] for more details. Our approach aims to sample a model, such as a neural network, rather than an output. Further, we aim to approximately sample from a posterior distribution rather than a distribution that generated observe data.

2 Training

Given data samples $\mathcal{D} = ((x_t, y_t) : t = 1, \ldots, T)$, a common approach to estimating the associated mapping between x and y entails selecting $\hat{\theta}$ to minimize a loss function of the form

$$\mathcal{L}(\hat{\theta}, \mathcal{D}) = \sum_{t=1}^{T} (f_{\hat{\theta}}(x_t) - y_t)^2 + \psi(\hat{\theta}).$$

However, we wish to sample $\hat{\theta}$ approximately from a posterior distribution of models conditioned on $\mathcal{D}$. Our general framework for doing this involves defining a perturbed loss function

$$\mathcal{L}_z(\hat{\theta}, \mathcal{D}) = \sum_{t=1}^{T} (f_{\hat{\theta}}(x_t, z) - y_t)^2 + \psi_z(\hat{\theta}) \tag{1}$$

and minimizing

$$\mathcal{L}(\hat{\theta}, \mathcal{D}) = \int_z p_z(dz) \mathcal{L}_z(\hat{\theta}, \mathcal{D}), \tag{2}$$

for some distribution p_z. Then, we sample a mapping from x to y by sampling $z \sim p_z$ and using $f_\theta(x, z)$.

2.1 Neural Networks

Let us now describe one example. Suppose $f_\theta : \Re^N \mapsto \Re$ represents a neural network with weights θ, for which the prior distribution is $\mathcal{N}(0, \sigma_0^2 I)$. Suppose we are given data pairs $\mathcal{D} = ((x_t, y_t) : t = 1, \ldots, T)$. Let

p_z be unit Gaussian over a K-dimensional space. In general, K will be much smaller than T. Let the rows of $B \in \Re^{N \times K}$ and a_t, for each $t = 1, \ldots, T$, be sampled uniformly at random from the K-dimensional unit sphere. Consider minimizing a loss function

$$\mathcal{L}(\hat{\theta}, \mathcal{D}) = \int_z p_z(dz) \left(\sum_{t=1}^{T} (f_{\hat{\theta}}(x_t, z) - y_t - a_t^\top z)^2 + \frac{1}{\sigma_0^2} \|\hat{\theta} - \tilde{\theta}\|_2^2 \right), \tag{3}$$

with $\tilde{\theta} = \sigma_0 B z$. Then, by sampling $z \sim N(0, I)$, $f_{\hat{\theta}}(\cdot, z)$ serves as an approximate posterior sample. Here, after fixing the index z, $f_{\hat{\theta}}(., z)$ can be considered a new neural network with weights as $\hat{\theta}(z)$.

As discussed in [5], regularizing as done in the above loss function can cause problems with the workings of neural network training procedures based on stochastic gradient descent. As such, it may be more effective to use a modified loss function of the form

$$\mathcal{L}(\hat{\theta}, \mathcal{D}) = \int_z p_z(dz) \left(\sum_{t=1}^{T} (f_{\hat{\theta}}(x_t, z) + g_{\tilde{\theta}}(x_t) - y_t - a_t^\top z)^2 \right) \tag{4}$$

Here, $g_{\tilde{\theta}}$ can be thought of as a random prior network.

2.2 Linear Regression

As a sanity check, we consider in this section a version of the approach specialized to linear regression. Examining this case offers intuition for how and why our approach can work.

Suppose that θ is an N-dimensional vector with prior distribution $N(\mu_0, \Sigma_0)$ and that, for each t, $y_t = \theta^T x_t + w_t$ with $w_t \sim \mathcal{N}(0, \sigma_w^2)$. As discussed in [2], minimizing the following perturbed loss function gives $\hat{\theta}$, a sample from the posterior distribution of θ conditioned on $\mathcal{D}$:

$$\mathcal{L}(\hat{\theta}, \mathcal{D}) = \sum_{t=1}^{T} \frac{1}{\sigma_w^2} (\hat{\theta}^T x_t - y_t - \tilde{w}_t)^2 + (\hat{\theta} - \tilde{\theta})^\top \Sigma_0^{-1} (\hat{\theta} - \tilde{\theta}),$$

where $\tilde{w}_t \sim \mathcal{N}(0, \sigma_w^2)$ and $\tilde{\theta} \sim \mathcal{N}(\mu_0, \Sigma_0)$. The resulting posterior sample $\hat{\theta}$ can be thought of as a function of $\tilde{\theta}$ and $(\tilde{w}_t : t = 1, \ldots, T\}$. Now suppose we want to sample approximately from the posterior in a way that depends on a random vector $z \sim N(0, I)$ of fixed dimension K rather than a random object with dimension growing with T. We can do this minimizing a loss function of the form

$$\mathcal{L}_z(\hat{\theta}, \mathcal{D}) \quad = \textstyle\sum_{t=1}^{T} \frac{1}{\sigma_w^2} (\hat{\theta}^T x_t - y_t - \sigma_w a_t^T z)^2 + (\hat{\theta} - \tilde{\theta})^\top \Sigma_0^{-1} (\hat{\theta} - \tilde{\theta}), \tag{5}$$

with $\tilde{\theta} = \mu_0 + \Sigma_0^{1/2} B z$. As a function of z, the optimal solution takes the form $\hat{\theta} = \delta + Mz$, for some $\delta \in \Re^N$ and $M \in \Re^{N \times K}$. Letting $f_{(\delta, M)}(x, z) = (\delta + Mz)^T x$, we can alternatively minimize

$$\mathcal{L}((\delta, M), \mathcal{D}) \quad = \textstyle\int_z p_z(dz) \sum_{t=1}^{T} \frac{1}{\sigma_w^2} ((\delta + Mz)^T x_t - y_t - \sigma_w a_t^T z)^2 + (\hat{\theta} - \tilde{\theta})^\top \Sigma_0^{-1} (\hat{\theta} - \tilde{\theta}), \tag{6}$$

to obtain the same values of δ and M. This is a special case of (3).

Note that the minimizing arguments of (6) are

$$\delta = \left(\frac{1}{\sigma_w^2} \sum_{t=1}^{T} x_t x_t^T + \Sigma_0^{-1} \right)^{-1} \left(\frac{1}{\sigma_w^2} \sum_{t=1}^{T} x_t y_t + \Sigma_0^{-1} \mu_0 \right), \tag{7}$$

$$M = \left(\frac{1}{\sigma_w^2} \sum_{t=1}^{T} x_t x_t^T + \Sigma_0^{-1} \right)^{-1} \left(\frac{1}{\sigma_w} \sum_{t=1}^{T} x_t a_t^T + \Sigma_0^{-1/2} B \right). \tag{8}$$

Lets consider C be a matrix with rows $a_1, a_2 \ldots, a_T$ concatenated with the rows of B. It is easy to see that if the rows of C are orthogonal then $\hat{\theta} = \delta + Mz$ is an exact sample from the posterior. However, we will generally want to use a fixed value of K that is much smaller than T, and as such, the rows of C will generally fail to be orthogonal. Nevertheless, this approach can serve to offer a useful approximation to a posterior sample.

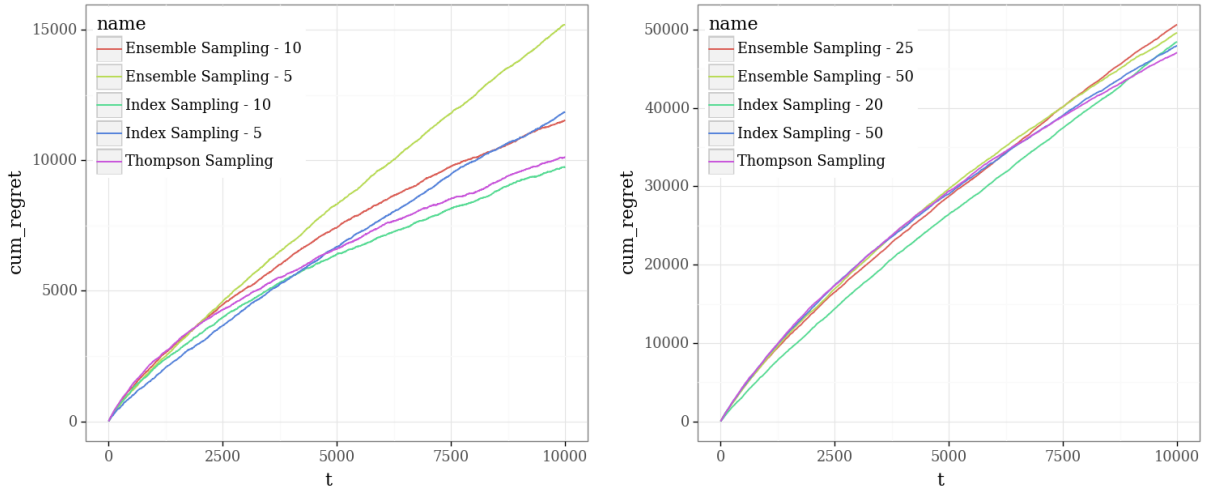
3 Computational Results

In Section 2.2, we have provided an intuition of why our method approximates sampling from posterior. In this section we will present some computational results to deepen the insight. We aim to paint a clearer picture which enhances our understanding than solving a difficult problem. In order to measure how well our estimated distribution approximates the posterior we look at it's performance in bandit settings where an agent picks an action from set of plausible actions and observe a stochastic reward corresponding to the chosen action. The agent aims to pick the best possible actions to maximize the cumulative reward. We compare our method which samples the model from an approximate posterior with Thompson sampling which samples from the actual posterior distribution. But for complicated models like neural networks obtaining posterior is very difficult if not computationally intractable. In such cases, we resort to comparing performance of our method with Ensemble sampling [2], which samples a model from a fixed size ensemble. Performance of an algorithm is quantified by regret(difference between maximum possible reward and reward obtained by the algorithm). For ease of convention we will be refering to our proposed method as Index sampling.

3.1 Gaussian Linear Bandit

We will first look at Gaussian linear bandits, an online linear optimization setting where rewards are given by bandit feedback where for each t, any action $x_t \in \Re^N$, $\|x_t\|_2 \leq 1$ the observed reward y_t is given by $x_t^T\theta + w_t$, where $w_t \sim \mathcal{N}(0, \sigma_w^2)$ is Gaussian noise with known variance σ_w^2 and $\theta \in \Re^N$ is the fixed unknown parameter which characterizes the model. Given a prior $\mathcal{N}(\mu_0, \Sigma_0)$ over θ, we initialize $\delta_1 = \mu_0$ and $M_1 = \Sigma_0^{1/2}B$. At any time t, we sample a model $\hat{\theta}_t$ from approximate posterior by picking a random index $z \in \Re^K$ and computing $\hat{\theta}_t = \delta_t + M_t z$. Now assuming that $\hat{\theta}_t$ is the true parameter we choose x_t which is optimal w.r.t $\hat{\theta}_t$ and observe y_t. Once observing (x_t, y_t), we update our model parameters δ, M incrementally using (7) and (8).

For comparing different methods we simulated a linear bandit with N dimensional input, with $\sigma_w^2 = 100$ and each coordinate of μ_0 is sampled uniformly from $[0, 10]$ and $\Sigma_0 = 10I_N$. Index sampling is implemented with index dimension as K and Ensemble sampling is implemented with M ensembles. Results are shown in the form of cumulative regret vs time on Figures 1a, 1b. Plots are obtained by averaging over 25 realizations for $N = 10$ and 5 realizations for $N = 50$.



(a) $N = 10$, Index Sampling with $K = 5$ and $K = 10$, Ensemble Sampling with $M = 5$ and $M = 10$

(b) $N = 50$, Index Sampling with $K = 20$ and $K = 50$, Ensemble Sampling with $M = 25$ and $M = 50$

Figure 1: Cumulative regret vs time for Index Sampling, Thompson Sampling and Ensemble Sampling

We observe that the index sampling method performs similar to that of Thompson sampling even for small values of K suggesting that the distribution from which we sample is approximately posterior. Performance of Index sampling when compared with ensemble sampling of $M \approx K$ is similar if not better.

3.2 Neural Network Bandit

Now we will consider a more complicated model, a 2-layer neural network bandit model for which obtaining posterior is difficult, computationally expensive. For the neural network bandit setting we compare performance of our method with ensemble sampling. Let the neural network be represented as $f_\theta : \Re^N \to \Re$ and at any time t, the observed output/reward for an action $x_t \in \mathcal{X} \subset \Re^N$ is given by $y_t = f_\theta(x_t) + w_t$ where $w_t \sim \mathcal{N}(0, \sigma_w^2)$ and θ are the unknown weights which define the neural network.

Our aim is to maximize the cumulative reward by picking actions from a finite set, $\mathcal{X}$. In case of index sampling, during inference a K-dimensional index is sampled and appended along with the input and passed to the neural network $f_{\hat{\theta}} : \Re^{N+K} \to \Re$, effectively sampling a f_θ from the approximate posterior.

We will consider a neural network $f_\theta(x) = W_2^T \max(0, W_1 x)$ with D hidden neurons and $\theta = (W_1, W_2)$ where $W_1 \in \Re^{N \times D}$, $W_2 \in \Re^D$ are weights of first and second layer respectively. For index sampling neural network is modified to $f_{\hat{\theta}}(x, z) = \hat{W}_2^T \max(0, \hat{W}_{11} x + \hat{W}_{12} z)$ where $\hat{W}_{11} \in \Re^{N \times D}$ and $\hat{W}_{12} \in \Re^{K \times D}$ are weights corresponding to first layer and $\hat{W}_2$ are weights of the second layer.

Experimental results are shown in Figure 2. In this experiment a neural network bandit with similar architecture as [2] simulated with input dimension $N = 100$, $D = 50$ hidden neurons, 100 actions are chosen uniformly at random from a unit box with last dimension set to 1, weights $W \sim \mathcal{N}(0, \sigma^2 I)$ with $\sigma = 1$ and $\sigma_w^2 = 100$. A learning rate of 0.01 and with a mini batch size of 64 samples are used while training. The results are averaged over 5 realizations.

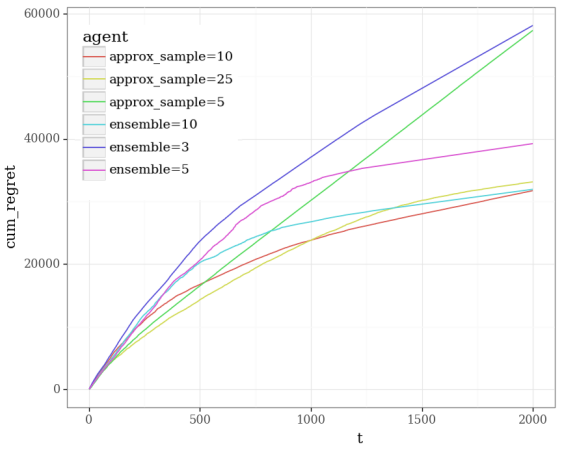


Figure 2: Cumulative regret for Index Sampling ($K = 5,\ 10,\ 25$) and Ensemble Sampling ($M = 3, 5, 10$)

We observe that even for $K = 25$, Index Sampling performs close to ensemble sampling with $M = 10$ even though index sampling has eight times fewer number of parameters. This resulted in a faster execution of index sampling saving both computation time and space.

References

[1] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on Thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

[2] Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. In *Advances in Neural Information Processing Systems*, pages 3258–3266, 2017.

[3] Ian Osband, Daniel Russo, Zheng Wen, and Benjamin Van Roy. Deep exploration via randomized value functions. *arXiv preprint arXiv:1703.07608*, 2017.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8626–8638, 2018.

[6] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[7] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

Self-improving Chatbots based on Reinforcement Learning

Elena Ricciardelli, Debmalya Biswas
AI Center of Excellence
Philip Morris International
Lausanne, Switzerland
firstname.lastname@pmi.com

Abstract

We present a Reinforcement Learning (RL) model for self-improving chatbots, specifically targeting FAQ-type chatbots. The model is not aimed at building a dialog system from scratch, but to leverage data from user conversations to improve chatbot performance. At the core of our approach is a score model, which is trained to score chatbot utterance-response tuples based on user feedback. The scores predicted by this model are used as rewards for the RL agent. Policy learning takes place offline, thanks to an user simulator which is fed with utterances from the FAQ-database. Policy learning is implemented using a Deep Q-Network (DQN) agent with epsilon-greedy exploration, which is tailored to effectively include fallback answers for out-of-scope questions.

The potential of our approach is shown on a small case extracted from an enterprise chatbot. It shows an increase in performance from an initial 50% success rate to 75% in 20-30 training epochs.

Keywords: Reinforcement Learning, Chatbots, NLP

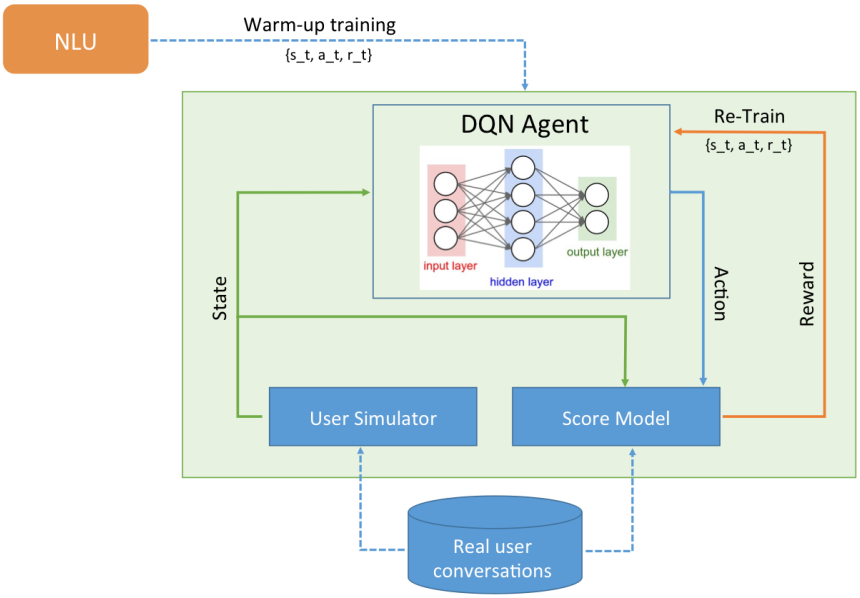


Figure 1: Architecture of the RL model used in this work. The DQN agent is initially trained offline in a warm-up phase on the NLU. The score model is also trained offline with the data from real user conversations. In the RL loop, the user state (user utterance) is provided by the user simulator, the action (chatbot response) is provided by the DQN agent and the reward is provided by the score model. Each tuple (s_t, a_t, r_t) feeds the experience replay buffer, which is used to re-train the DQN after $n_{episodes}$ episodes, which is a tunable parameter.

1 Introduction

The majority of dialog agents in an enterprise setting are domain specific, consisting of a Natural Language Understanding (NLU) unit trained to recognize the user's goal in a supervised manner. However, collecting a good training set for a production system is a time-consuming and cumbersome process. Chatbots covering a wide range of intents often face poor performance due to intent overlap and confusion. Furthermore, it is difficult to autonomously retrain a chatbot taking into account the user feedback from live usage or testing phase. Self-improving chatbots are challenging to achieve, primarily because of the difficulty in choosing and prioritizing metrics for chatbot performance evaluation. Ideally, one wants a dialog agent to be capable to learn from the user's experience and improve autonomously.

In this work, we present a reinforcement learning approach for self-improving chatbots, specifically targeting FAQ-type chatbots. The core of such chatbots is an intent recognition NLU, which is trained with hard-coded examples of question variations. When no intent is matched with a confidence level above 30%, the chatbot returns a fallback answer. For all others, the NLU engine returns the corresponding confidence level along with the response.

Several research papers [2, 3, 7, 8] have shown the effectiveness of a RL approach in developing dialog systems. Critical to this approach is the choice of a good reward model. A typical reward model is the implementation of a penalty term for each dialog turn. However, such rewards only apply to task completion chatbots where the purpose of the agent is to satisfy user's request in the shortest time, but it is not suitable for FAQ-type chatbots where the chatbot is expected to provide a good answer in one turn. The user's feedback can also be used as a reward model in an online reinforcement learning. However, applying RL on live conversations can be challenging and it may incur a significant cost in case of RL failure. A better approach for deployed systems is to perform the RL training offline and then update the NLU policy once satisfactory levels of performance have been reached.

2 Reinforcement Learning Model

The RL model architecture is illustrated in Figure 1. The various components of the model are: the NLU unit, which is used to initially train the RL agent in a warm-up phase; the user simulator, which randomly extracts the user utterances from the database of user experiences; the score model trained on the user's conversation with feedback and the RL agent based on a Deep Q-Network (DQN) network.

2.1 Dialog System

We apply the reinforcement learning approach on a FAQ-type chatbot. At the core of the chatbot, there is an intent[1] recognition NLU, which is trained with hard-coded examples of question variations. For this work, we have chosen the open-source NLU from Rasa[2], using the TensorFlow pipeline. However the RL approach is independent from the NLU chosen and for systems in production it can easily be extended to NLU engines such as IBM Watson or Amazon LEX.

2.2 Real User Conversations

We used user feedback obtained during the development an actual internal chatbot for our work. The scope of the chatbot was to answer employee queries related to office building facilities, HR policies and benefits, etc. All the 10 users participating in the test phase were informed that their feedback would be used to improve the chatbot performance. The testers provided a (binary) feedback after each conversation turn, thus rating the utterance-response tuples. The historical data thus contains quadruples of the following format: (utterance, response, NLU confidence level and feedback). By removing non valid conversations (i.e. those lacking or with invalid feedback) we end up with 550 user conversations, triggering about 120 intents. Although we have tested the score model on all the conversations, the RL model has been applied only on a subsample of 155 conversations, triggering the top 5 intents. On this subset, the user's satisfaction is 50%.

Table 1: Example of conversation from the database, as well as the score provided by the model and by the user

Utterance	**Response**	**Model score**	**User feedback**
What is Interact?	I can help you mainly with GS service-related questions on topics like transportation (your mobility plan/benefit car), business travel, hospitality and events, gym, mailroom etc.	0.30	0
where can I find my salary payment	I was getting bored, but now you are here.	0.26	0
Have a nice day	Bye bye! Thanks for the chat!	0.81	1

2.3 Reward Function: the Score Model

Evaluating chatbot performance is a long-standing issue in computational linguistic. Automatic metrics borrowed from machine translations (e.g. [6]) do not perform well on short sentences (e.g. [4]), such as the chatbot utterance-response tuples. On the other hand, human rating of chatbots is by now the *de-facto* standard to evaluate the success of a chatbot, although those ratings are often difficult and expensive to gather.

To evaluate the correctness of chatbot responses, we propose a new approach which makes use of the user conversation logs, gathered during the development and testing phases of the chatbot. Each user had been asked to provide a binary feedback (positive/negative) at each chatbot turn. In order to use the user feedback in an offline reinforcement learning, we have developed a score model, capable of modeling the binary feedback for unseen utterance-response tuples. In a supervised fashion, the score model learns how to project the vector representations of utterance and response in a linearly transformed space, such that similar vector representations give high score. As for the vector representation of sentences, we compute sentence embedding through the universal sentence encoder [1], available through TensorFlow Hub [3]. To train the model, the optimization is done on a squared error (between model prediction and human feedback) loss with L2 regulation. To evaluate the model, the predicted scores are then converted into a binary outcome and compared with the targets (the user feedbacks). For those couples of utterances having a recognized intent with both a positive feedback and a NLU confidence level close to 1, we perform data augmentation, assigning low scores to the combination of utterance and fallback intent.

A similar approach for chatbot evaluation has been suggested by [4]. The authors model the scores by using a labelled set of conversations, that also include model and human-generated responses, collected through crowdsourcing. Our approach differs from the above authors in that it just requires a labelled set of utterance-response tuples, which are relatively straightforward to gather during the chatbot development and user testing phases.

[1] An intent is defined as the user's intention, which is formulated through the utterance

[2] https://rasa.com/

[3] https://www.tensorflow.org/hub

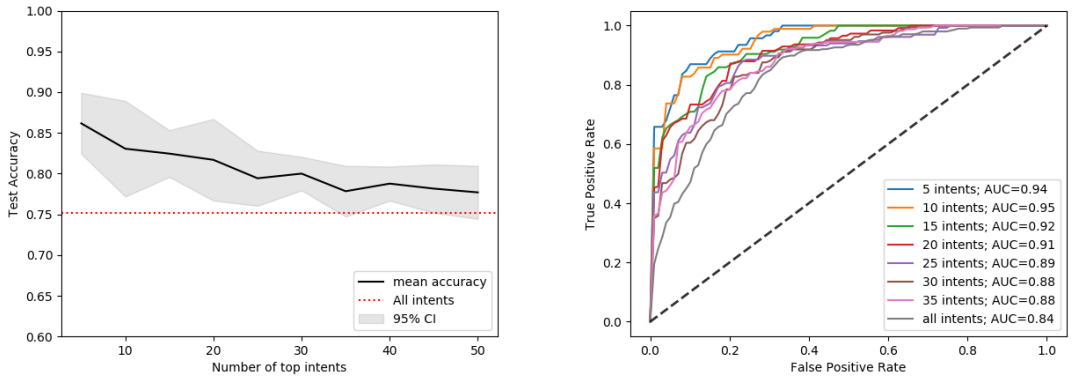


Figure 2: Performances of the score model. Left-hand panel: cross-validated test set accuracy with 95% confidence interval for different sub-samples having different number of intents. The horizontal red line indicates the performances for the entire sample. Right-hand panel: ROC curves for the different subsamples.

2.4 Policy Learning with DQN

To learn the policy, the RL agent uses a Q-learning algorithm with DQN architecture [5]. In DQN, a neural network is trained to approximate the state-action function $Q(s_t|a_t, \theta)$, which represents the quality of an action a_t provided a state s_t, and θ are the trainable parameters. As for the DQN network, we have followed the approach proposed by [3], using a fully-connected network, fed by an experience replay pool buffer, that contains the one-hot representation of utterance and response and the corresponding reward. An one-hot representation is possible in this case as we have a finite possible values for utterances (given by the number of real users's question in the logs) and responses (equal to the number of intents used on out test-case, 5). In a warm-up phase, the DQN is trained on the NLU, using as a reward the NLU confidence level. The DQN training set is augmented whenever a state-action pair has a confidence above a threshold, by assigning zero weight to the given state and all the other available actions. Thus, at the starting of the RL training, the agent performs similar to the NLU unit.

During RL training, we use an ϵ-greedy exploration, where random actions are explored according to a probability ϵ. We use a time-varying ϵ which facilitates the exploration at the beginning of the training with $\epsilon_{t_0} = 0.2$ and $\epsilon_t = 0.05$ during the last epoch. To speed-up the learning when picking random actions, we also force higher probability to get a "No intent detected", as several questions are actually out of the chatbot scope, but they are erroneously matched to a wrong intent by the NLU. During an epoch we simulate a batch of conversations of size $n_{episodes}$ (ranging from 10 to 30 in our experiments) and fill an experience replay buffer with the tuple (s_t, a_t, r_t). The buffer has fixed size and it is flushed the first time when the agent performance increases above a specified threshold. In those episodes where the state-action tuple gets a reward greater than 50%, we perform data augmentation by assigning zero reward to the assignment of any other action to the current state.

3 Model Evaluation

3.1 Score Model Evaluation

To evaluate the model, we select subsets of conversations, triggering the top N intents, with N between 5 and 50. The results of the score model are summarized in Figure 2, showing the cross-validated (5-*fold* CV) accuracy on the test set and the ROC curve as a function of the number of intents. For the whole sample of conversations, we obtain a cross-validated accuracy of 75% and an AUC of 0.84. However, by selecting only those conversations triggering the top 5 intents, thus including more examples per intent, we obtain an accuracy of 86% and an AUC of 0.94. For the RL model evaluation, we have focussed on the 5 intents subsets; which ensures that we have the most reliable rewards.

3.2 Reinforcement Learning Model Evaluation

The learning curve for the RL training is shown in Figure 3. In the left-hand panel, we compare the RL training with the reward model with a test done with a direct reward (in interactive way), showing that the score model is giving similar performances to the reference case, where the reward is known. Large fluctuations in the average score are due to a limited batch size ($n_{episodes} = 10$) and a relatively large ϵ. We also show the success rate on a test set of 20 conversations, extracted from the full sample, where a "golden response" is manually provided for all the utterances. The agent success rate increases from an initial $\sim 50\%$ to 75% in only ~ 30 epochs, showing the potential of this approach. In the right-hand panel, we show the results using $n_{episodes} = 30$, showing similar performances but with a smoother learning curve.

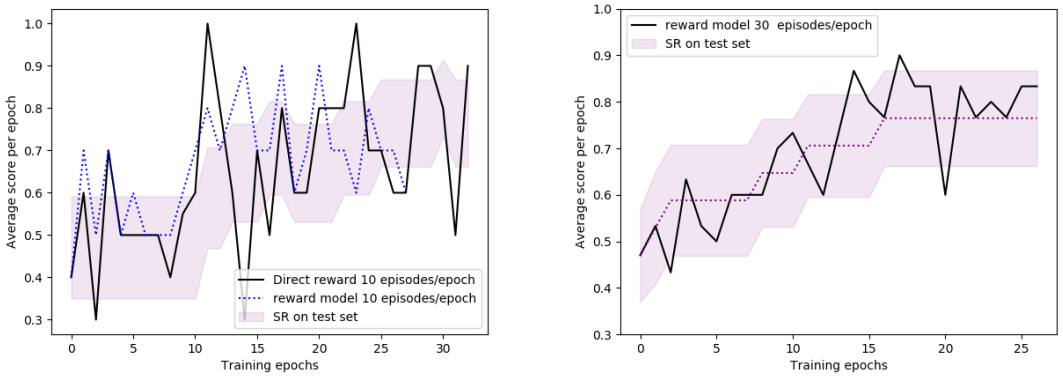


Figure 3: Learning curves showing the DQN agent's average score (continuous black line) per training epoch and success rate (purple shaded area) based on a labelled test set of 20 conversations. Left-hand panel: learning curves for direct RL with interactive reward (black line) and the reward model (blue dotted line), using 10 episodes per epoch. Right-hand panel: learning curves for the model reward, using 30 episodes per epoch.

4 Conclusions

In this work, we have shown the potential of a reinforcement learning approach in improving the performance of FAQ-type chatbots, based on the feedback from a user testing phase. To achieve this, we have developed a score model, which is able to predict the user's satisfaction on utterance-response tuples, and implemented a DQN reinforcement model, using the score model predictions as rewards. We have evaluated the model on a small, but real, test case, demonstrating promising results. Further training on more epochs and including more data, as well as extensive tests on the model hyper-parameters are in progress. The value of our approach is in providing a practical tool to improve large-scale chatbots (with a large set of diverse intents), in an automated fashion based on user feedback.

Finally, we notice that although the reinforcement learning model presented in this work is suitable for FAQ-type chatbots, it can be generalised to include the sequential nature of conversation by incorporating a more complex score model.

References

[1] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[2] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[3] Xiujun Li, Yun-Nung Chen, Jianfeng Gao, and Asli Celikyilmaz. End-to-end task-completion neural dialogue systems. In *8th International Joint Conference on Natural Language Processing*, 2017.

[4] Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126. Association for Computational Linguistics, 2017.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, 02 2015.

[6] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, 2002.

[7] Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2182–2192. Association for Computational Linguistics, 2018.

[8] Iulian Vlad Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Mudumba, Alexandre de Brébisson, Jose Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, and Yoshua Bengio. A deep reinforcement learning chatbot. *CoRR*, abs/1709.02349, 2017.

Modeling models of others' mental states: characterizing Theory of Mind during cooperative interaction

Tessa Rusch
Inst. for Systems Neuroscience
University Hamburg
t.rusch@uke.de

Prashant Doshi
Dept. of Computer Science
University of Georgia
pdoshi@cs.uga.edu

Martin Hebart
National Inst. of Mental Health
Bethesda
martin.hebart@nih.gov

Saurabh Kumar
Inst. for Systems Neuroscience
University Hamburg
s.kumar@uke.de

Michael Spezio
Psychology & Neuroscience, Scripps College
Inst. for Systems Neuroscience, University Hamburg
mspezio@scrippscollege.edu

Jan Gläscher
Inst. for Systems Neuroscience
University Hamburg
glaescher@uke.de

Abstract

Humans are experts in cooperation. To effectively engage with others they have to apply Theory of Mind (ToM), that is they have to model others beliefs, desires, and intentions and predict their behavior from these mental states. Here, we investigate ToM processes during real-time reciprocal coordination between two players engaging in a cooperative decision game. The game consists of a noisy and unstable environment. To succeed participants have to model the state of the world and their partner's belief about it and integrate both pieces of information into a coherent decision. Thereby the game combines social and non-social learning into a single decision problem. To quantify the learning processes underlying participants' actions, we modeled the behavior with Interactive Partially Observable Markov Decisions Processes (I-POMDP). The I-POMDP framework extends single agent action planning under uncertainty to the multi-agent domain by including intentional models of other agents. Using this framework we successfully predicted interactive behavior. Furthermore, we extracted participants' beliefs about the environment and their beliefs about the mental states of their partners, giving us direct access to the cognitive operations underling cooperative behavior. By relating players' own beliefs with their partners' model of themselves we show that dyads whose beliefs are more aligned coordinate more successfully. This provides strong evidence that behavioral coordination relies on mental alignment.

Keywords: Theory of Mind, mentalizing, state uncertainty, cooperation, modeling

Acknowledgements

J. G. was supported by the Bernstein Award for Computational Neuroscience (BMBF 01GQ1006) and J.G. and M.S. were supported by a Collaborative Research in Computational Neuroscience (CRCNS) grant (BMBF 01GQ1603; NSF 1608278). T.R. was supported by a PhD scholarship from the German National Merit Foundation.

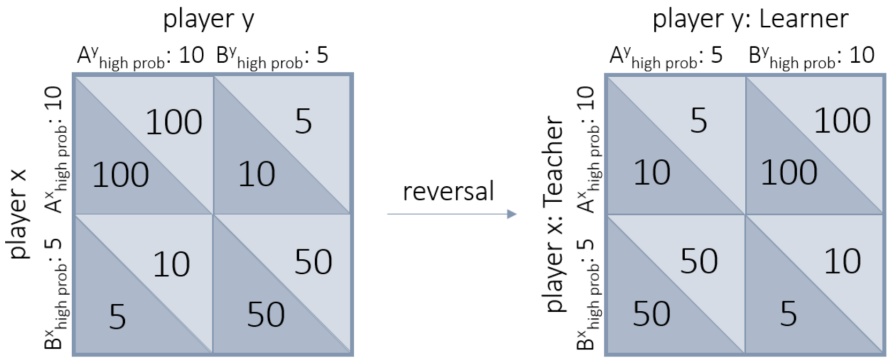
Extended Abstract

Cooperation is the capacity to act in accordance with the perceptions, goals, and beliefs of others to facilitate the own and other's gain equitably. Humans are distinctly skilled at this capacity. Examples encompass a teacher passing on knowledge to a student, or engineers working together to develop self-driving cars or new means of producing renewable energy. Cognitively, cooperation without prior coordination often requires Theory of Mind (ToM), i.e. the capacity to estimate and represent others' mental states and predict rational behavior based on these mental states. To successfully cooperate, humans have to combine the predictions of their partners' behavior with their knowledge of the world and act according to the combined requirements of the interactive situation. Here we set out to investigate the cognitive processes allowing humans to cooperate in a formalized and quantifiable way. We aim at capturing the models that humans build of others' mental states and the world and how they integrate these pieces of information to produce coordinated actions.

To examine the processes underlying cooperative behavior in a truly interactive setting we developed a novel decision-making task that requires participants to make cooperative choices on probabilistic and occasionally changing options. The reward structure of the task facilitates cooperation by incentivizing identical joint actions by both partners. This is complicated by the facts that participants receive only noisy observations of the underlying reward structure and that the players' reward contingencies can reverse unpredictably. Additionally, one player has more knowledge about the reward contingencies than the other player. The asymmetry of knowledge between two agents resembles the situation in the classic false-belief-task (Wimmer & Perner, 1983), in which the (all-knowing) participant has to realize that the girl in the story has a false belief about the environment and therefore makes an incorrect choice. In our setup, the less informed player has a false belief and the informed player a correct belief about the world. This divergent knowledge prompts participants to observe and learn about the reward contingencies, but also to model and track the partner's belief about the world (as well as their partner's belief about themselves) because the reward is maximal, when both players coordinate their actions. Therefore, they need to bring their world knowledge and their model of the partner's mental state together into a single valuation process.

To gain access to the private cognitive operations that allow humans to coordinate in a complex setting, we model the behavioral data in the context of the I-POMDP framework (Gmytrasiewicz & Doshi, 2005). I-PODMPs extend single-agent action planning in an uncertain environment to the interactive domain by including intentional models of other agents that themselves engage in action planning. These models of others may themselves include models of the original agent allowing the capture of recursive reasoning processes humans can engage in during strategic interaction.

Task Details

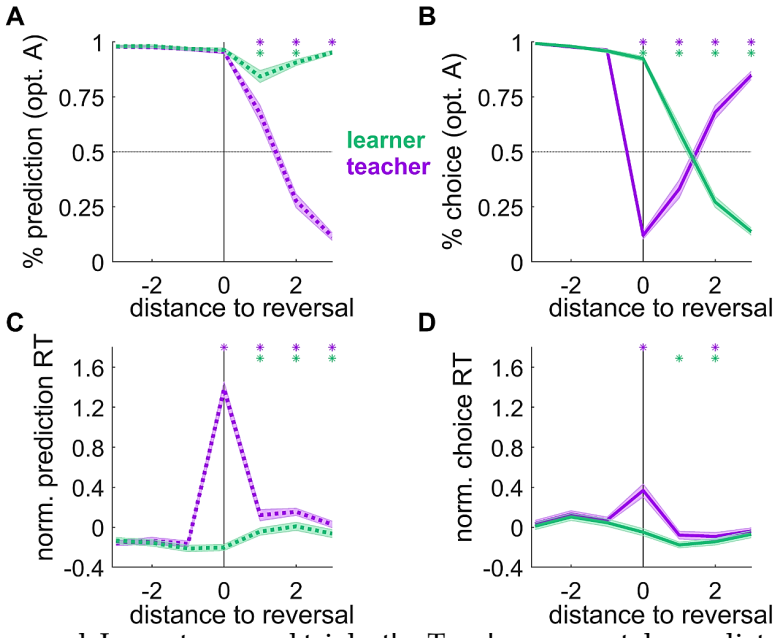


The task used here extends the concept of the classic false belief task to the interactive domain. We therefore refer to it as the "Interactive False Belief Task" (IFBT). In the IFBT two players choose between two options ("left" or "right") for probabilistic rewards. One option has a high probability for a high reward (10), the other a high probability for a low reward (5). Using trial and error the participants have to figure whether the high reward is on the left or on the right. When both partners obtain the same individual outcome, they are rewarded by a ten-fold increase of their individual outcomes. If individual outcomes differ, they receive the nominal individual outcomes. Their own reward distribution and the partner's action are unknown to the players, but have to be inferred form the received outcome. The partner's reward distribution is openly presented to the players at the beginning of each trial. Prior to their own choice, participants have to predict the partner's action. In the displayed reward matrices, the initial setting is shown on the left. Both players need to choose option "A" to receive the individual high outcome. Thereby, the probability of receiving the maximum reward of 100/100 is highest. However, due to the probabilistic choice-outcome

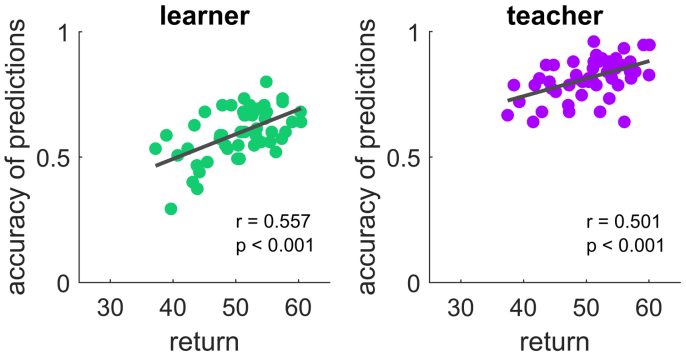
relation, all other outcomes are also possible. After a few trials, one player's (here: Player y's) reward contingencies are reversed, i.e. this player's high option moves from left to right or vice versa, while the partner's reward contingencies remain the same. This player remains uninformed about the change and is therefore referred to as the "Learner". As Learners are ignorant to reversals, they hold a false belief about the reward structure of the task. The partner is informed about the contingency reversal, hence we call this player "Teacher". For the rest of this abstract, for ease of reference we will refer to the Teacher as "she" and the Learner as "he". This is unrelated to the participants' gender, as we tested an equal number of male and female participants and all participants played both roles (total N = 50, 25 female). Taking the Learner's false belief into account the Teacher has to choose the less valuable option "B" at the reversal. The most likely ensuing reward of 50/50 signals the Learner that his reward contingencies have reversed. After a period of stable coordination, reversals repeat. Throughout the game reversals are unpredictable and players are randomly assorted to the roles of Teacher and Learner. Thereby, participants have to stay attentive at all times.

Model-free analysis

The Learner's main task in the IFBT is to detect and react to changes in the reward-contingencies. The Teacher is fully informed about the change. Her goal is to "communicate" these reversals through her choices. She has to react to the Learner's decisions at the reversal and to his choice adaptation after the reversal.



Players' predictions about the partner's choices are shown in the (A), players' own choices in (B). The respective response times (RTs) are shown in (B) and (C). Model free analyses show, that the Learner detects reversals and gradually shifts his choices after a reversal. During the reversal, the Teacher correctly predict that the Learner stays with his previous choice but switches her own choice to "B". Even though the Teacher's prediction is identical to his pre-reversal prediction, her RTs are 6-fold increased (purple RTs curve in (C)) at reversal. In post reversal trials, the Teacher accurately predicts the partner's choice curve (purple prediction curve in (A) and green choice curve in (B)) and matches the Learner's choice switching by returning to her pre-reversal choice at the same rate (purple choice curve in (B)). During this period, the Teacher's prediction RTs remain elevated. These results strongly suggest, that participants actively engage in mentalizing to solve the task. Further, the response times show that mentalizing is a computationally demanding cognitive process.



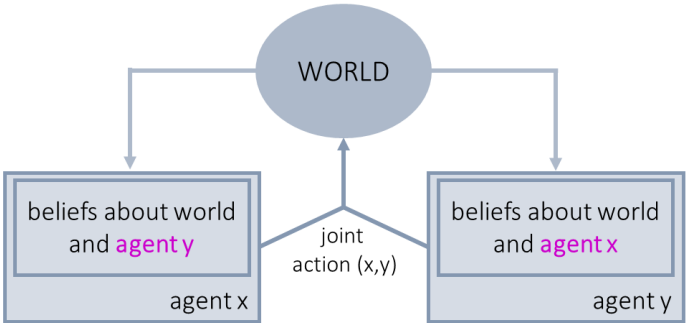
The beneficial effect of mentalizing is shown in a strong correlation between players' accuracy in predicting their partner's choice and their average obtained return. This entails that successful mentalizing facilitates cooperative behavior.

Computational Modeling with the I-POMDP

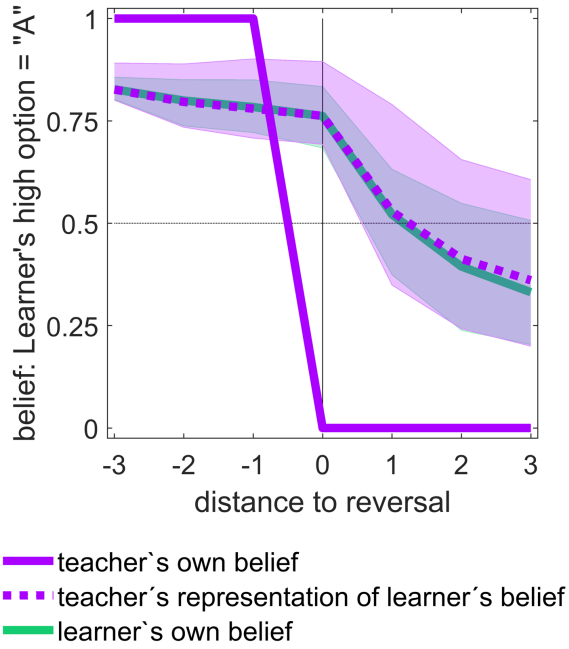
In the IFBT outcomes are probabilistically associated with the participants' choice options. The goal of the task is to maximize the joint outcome, which by design of the payoff matrix is identical to maximizing the individual reward. To achieve this goal the participants have to generate beliefs about which option is currently the best. Using their actions and the observations of the resulting joint outcome they can update this belief distribution. After a reversal, the Learner does not know that his state has changed. His belief is

therefore false. The Teacher is aware of the state change. Further, she is informed that the Learner does not know about the reversal. Thereby, she can infer the Learner's false belief, correctly predict the Learner's (wrong) action, and accommodate for it by switching her own choice. Previous studies examining ToM in interactive tasks did not include state uncertainty (Devaine, Hollard, & Daunizeau, 2014; Hill et al., 2017; Yoshida, Dolan, & Friston, 2008). In these studies, representing another persons' beliefs is unnecessary, as in a fully and perfectly observable world, others' beliefs about the states should be identical to one's own belief. In this study, however, participants interacted in a highly uncertain environment. Therefore, we need to address the attribution of beliefs to others, a core component of ToM.

Single agent action planning under uncertainty is well captured by partially observable Markov Decision Processes (POMDPs) (Kaelbling, Littman, & Cassandra, 1998). The innovative element of POMDPs is that the agent maintains a belief about which states he could be in. Beliefs are represented by probability distributions over all possible states. At each time step the agent's belief is updated with a Bayesian learning rule. In the context of the IFBT, states are specified by the location of the high reward option (possible states are "High Left (HL)" and "High Right (HR)"). Here, we extend the problem to the multi-agent domain. To capture humans mentalizing during interaction in an uncertain environment, we apply *Interactive* POMDPs (I-POMDPs) (Gmytrasiewicz & Doshi, 2005), graphically illustrated on the right. In contrast to single agent POMDPs, I-POMDPs contain an agent's belief about the states of the world *and* a belief about the mental states of the other agent, which is the other agent's belief about the states of the world. For the IFBT this means that agents form a belief about the location of their own and their partner's high option, and about the partner's belief about the distribution of rewards. As in the single agent model, beliefs are updated at each time step. In the multi-agent framework, the belief about the other's mental state is updated by simulating the partner's learning process. These core features of the framework make it an ideal candidate for modeling participants' behavior in the IFBT and access the underlying critical belief computations.

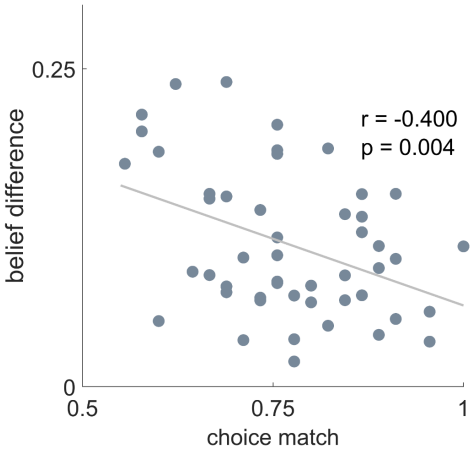


We fitted parametrized I-POMDP models to behavioral data from the IFBT and found that I-POMDPs predict the Teacher's and the Learner's actions with high accuracy. The Teacher's and Learner's beliefs about the Learner's reward contingencies (HL/HR) as computed by fitted I-POMDPs are shown on the right. The Teacher's own belief about the Learner's reward contingencies is represented by the solid purple line. The Teacher is fully informed and therefore knows, that the Learner's reward contingencies are reversed at trial 0. This is reflected in the sudden drop in the curve illustrating the Teacher's belief about the Learner's high option. The Learner's own belief about his rewards is shown in the solid green line (partially hidden beneath the dotted purple line). Before a reversal, he correctly represents the state of the task. At the reversal, his belief becomes false, but he is not aware of it yet. The shift in the Learner's belief only starts in the post-reversal period reflecting gradual adaptation to the change in reward contingencies. The Teacher's belief about the Learner's belief is illustrated by the dotted purple line. It matches the Learner's own beliefs almost perfectly. This shows that Teachers accurately represent their partner's mental state.



In a subsequent step we examine the consequences of these beliefs on the coordination of behavior. In the IFBT, successful cooperation is achieved by a matching of choices between the two players of a dyad. We find

that in dyads, in which the Teacher's representation of the Learner's belief is more similar to the Learner's own belief (i.e., the belief difference is small), choice matching is more pronounced as illustrated in the significant negative correlation on the right. These results suggest that successful cooperation is enhanced by a coordination of mental states.



Conclusion

Successful cooperation often requires the coordination of joint actions. Here we provide behavioral and computational evidence that humans represent their partners as rational intentional agents and model their mental states. Using the I-POMDP framework we can formalize and quantitatively estimate these Theory of Mind processes. Our modeling findings suggests that humans incorporated mental models of their partners into their own model of the world and use it to guide coherent decision making leading to successful cooperation through mental coordination.

References

Devaine, M., Hollard, G., & Daunizeau, J. (2014). The Social Bayesian Brain: Does Mentalizing Make a Difference When We Learn? *PLoS Computational Biology*, *10*(12). http://doi.org/10.1371/journal.pcbi.1003992

Gmytrasiewicz, P. J., & Doshi, P. (2005). A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, *24*, 49–79. http://doi.org/10.1613/jair.1579

Hill, C. A., Suzuki, S., Polania, R., Moisa, M., Doherty, J. P. O., & Ruff, C. C. (2017). A causal account of the brain network computations underlying strategic social behavior, *20*(8). http://doi.org/10.1038/nn.4602

Kaelbling, L., Littman, M., & Cassandra, A. (1998). Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, *101*(1–2), 99–134. http://doi.org/10.1016/S0004-3702(98)00023-X

Wimmer, H., & Perner, J. (1983). Beliefs about beliefs: Representation and constraircing function of wrong bekfs in young children's understanding of deception. *Cognition*, *13*(1), 103–128.

Yoshida, W., Dolan, R. J., & Friston, K. J. (2008). Game theory of mind. *PLoS Computational Biology*, *4*(12), e1000254. http://doi.org/10.1371/journal.pcbi.1000254

Learning from Suboptimal Demonstrations: Inverse Reinforcement Learning from Ranked Observations

Daniel S. Brown*
Department of Computer Science
The University of Texas at Austin
dsbrown@cs.utexas.edu

Wonjoon Goo*
Department of Computer Science
The University of Texas at Austin
wonjoon@cs.utexas.edu

Prabhat Nagarajan
Preferred Networks
Tokyo, Japan
prabhat@preferred.jp

Scott Niekum
Department of Computer Science
The University of Texas at Austin
sniekum@cs.utexas.edu

Abstract

A critical flaw of existing imitation learning and inverse reinforcement learning methods is their inability, often by design, to significantly outperform the demonstrator. This is a consequence of the general reliance of these algorithms upon some form of mimicry, such as feature-count matching or behavioral cloning, rather than inferring the underlying intentions of the demonstrator that may have been poorly executed in practice. In this paper, we introduce a novel reward-learning-from-observation algorithm, Trajectory-ranked Reward EXtrapolation (T-REX), that extrapolates beyond a set of ranked suboptimal demonstrations in order to infer a high-quality reward function. We leverage the pairwise preferences induced from the ranked demonstrations to perform reward learning without requiring an MDP solver. By learning a state-based reward function that assigns greater return to higher-ranked trajectories than lower-ranked trajectories, we transform a typically expensive, and often intractable, inverse reinforcement learning problem into one of standard binary classification. Moreover, by learning a reward function that is solely a function of state, we are able to learn from observations alone, eliminating the need for action labels. We combine our learned reward function with deep reinforcement learning and show that our approach results in performance that is better than the best-performing demonstration on multiple Atari and MuJoCo benchmark tasks. In comparison, state-of-the-art imitation learning algorithms fails to exceed the average performance of the demonstrator.

Keywords: Inverse Reinforcement Learning, Learning from Observations, Learning from Ranked Demonstrations, Suboptimal Demonstrations, Reward Learning

Acknowledgements

This work has taken place in the Personal AutonomousRobotics Lab (PeARL) at The University of Texas at Austin. PeARL research is supported in part by the NSF (IIS-1724157, IIS-1638107, IIS-1617639, IIS-1749204) and ONR(N00014-18-2243).

*Equal contribution

1 Introduction

When goals or rewards are difficult for a human to specify, inverse reinforcement learning (IRL) [1] techniques can be applied to infer the goals of a user from demonstrations. Unfortunately, high-quality demonstrations are often difficult to provide for many tasks—for instance, consider a non-expert user attempting to give kinesthetic demonstrations of a household chore to a robot. Even for human experts, tasks such as high-frequency stock trading, complex video games, or sports involving fine motor skills can be virtually impossible to perform optimally.

If a demonstrator is suboptimal, but his intentions can be ascertained, then in principle, a learning agent ought to be able to exceed the demonstrator's performance. However, current IRL algorithms fail to do this, typically searching for a reward function that makes the demonstrations appear near-optimal [1]. When demonstrations are suboptimal, these methods are, by design, unable to learn a policy that is significantly better than the demonstrator. Imitation learning approaches that mimic behavior directly, such as behavioral cloning [8], also suffer from this limitation.

To overcome this critical flaw in current imitation learning methods, we propose a novel IRL algorithm, Trajectory-ranked Reward EXtrapolation (T-REX) that utilizes a ranking amongst the demonstrations to extrapolate a user's underlying intent beyond the best demonstration. This, in turn, enables a reinforcement learning agent to exceed the performance of the demonstrator by learning to optimize this extrapolated reward function. Specifically, we use ranked demonstrations to learn a state-based reward function that assigns greater total return to higher-ranked trajectories. Learning a reward function from ranked demonstrations in this way has three key advantages: (1) rather than imitating suboptimal demonstrations, it allows us to identify whether features are positively or negatively correlated with the ground-truth reward, allowing for better-than-expert performance; (2) by leveraging the pairwise preferences contained in the ranked demonstrations, reward learning becomes a standard binary classification problem, without the need to repeatedly solve, or partially solve, an MDP; and (3) by learning a reward function that is solely a function of state, we are able to learn a reward function from observations without requiring access to the demonstrator's actions.

Our work builds on recent work on reward learning through active queries [3] and is similar to recent work which combines an initial set of good demonstrations together with active preference queries to learn a reward function [6]. However, we explore the problem of extrapolating from ranked suboptimal demonstrations, rather than learning a reward from active queries during policy training.

We evaluate T-REX on a variety of standard Atari and MuJoCo benchmark tasks. Our experiments show that T-REX is capable of good extrapolation, often outperforming the best demonstration, as well as significantly outperforming state-of-the-art imitation learning and IRL algorithms that learn from observations.

2 Problem Definition

We model the environment as a Markov decision process (MDP) consisting of a set of states $\mathcal{S}$, actions $\mathcal{A}$, transition probabilities T, reward function r, and discount factor γ. A policy is a mapping from states to probabilities over actions, $\pi(a|s) \in [0, 1]$. The discounted expected return of a policy is given by $J(\pi) = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_t|\pi]$. We are concerned with the problem of inverse reinforcement learning from observation, where we do not have access to the reward function of the MDP. Instead, an agent is given a set of demonstrations $\mathcal{D}$ consisting of trajectories (sequences of states) from which it seeks to recover the reward function that the demonstrator is attempting to optimize. Given a sequence of m ranked trajectories τ_t for $t = 1, \ldots, m$, where $\tau_i \prec \tau_j$ if $i < j$, we wish to find a parameterized reward function, $\hat{r}_\theta$, that approximates the true reward function r. Given $\hat{r}_\theta$, we then seek to optimize a policy $\hat{\pi}$ that can outperform the best demonstration via reinforcement learning on $\hat{r}_\theta$.

3 Methodology

We now describe Trajectory-ranked Reward EXtrapolation (T-REX)[1], an algorithm for using ranked suboptimal demonstrations to extrapolate a user's underlying intent beyond the demonstrations (see [2] for the full version of this paper). Given a sequence of m demonstrations ranked from worst to best, $\tau_1, \ldots, \tau_m$, T-REX has two steps: (1) reward inference and (2) policy optimization.

Given the ranked demonstrations, T-REX performs reward inference by approximating the reward at state s using a neural network, $\hat{r}_\theta(s)$, such that $\sum_{s\in\tau_i} \hat{r}_\theta(s) < \sum_{s\in\tau_j} \hat{r}_\theta(s)$ when $\tau_i \prec \tau_j$. The parameterized reward function $\hat{r}_\theta$ can be trained with ranked demonstrations using the generalized loss function:

$$\mathcal{L}(\theta) = \mathbf{E}_{\tau_i,\tau_j\sim\Pi}\Big[\xi\Big(\mathrm{P}\big(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)\big), \tau_i \prec \tau_j\Big)\Big], \tag{1}$$

[1]Code available at https://github.com/hiwonjoon/ICML2019-TREX

where Π is a distribution over demonstrations, ξ is a binary classification loss function, $\hat{J}$ is the cumulative return from the parameterized reward function $\hat{r}_\theta$, and $\prec$ is an indication of the preference between the demonstration trajectories.

We specifically represent the probability P as a softmax-distribution and we represent ξ using a cross entropy loss:

$$\mathrm{P}\big(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)\big) \approx \frac{\exp \sum_{s\in\tau_j} \hat{r}_\theta(s)}{\exp \sum_{s\in\tau_i} \hat{r}_\theta(s) + \exp \sum_{s\in\tau_j} \hat{r}_\theta(s)}, \quad \mathcal{L}(\theta) \approx -\sum_{\tau_i \prec \tau_j} \log \frac{\exp \sum_{s\in\tau_j} \hat{r}_\theta(s)}{\exp \sum_{s\in\tau_i} \hat{r}_\theta(s) + \exp \sum_{s\in\tau_j} \hat{r}_\theta(s)}. \tag{2}$$

This loss function trains a classifier that can predict whether one trajectory is preferable to another based on the predicted returns of each trajectory, and has been used previously for reward inference in active learning settings [3, 6].

To increase the number of training examples, T-REX trains on partial trajectory pairs rather than full trajectory pairs. This results in noisy preference labels that are only weakly supervised; however, using data augmentation to obtain pairwise preferences over many partial trajectories allows T-REX to learn expressive neural network reward functions from only a small number of ranked demonstrations. Given the learned reward function $\hat{r}_\theta(s)$, T-REX then seeks to optimizes a policy $\hat{\pi}$ with better-than-demonstrator performance through reinforcement learning.

4 Experiments and Results

We evaluated T-REX on three simulated robotic locomotion tasks and eight Atari games. We used the OpenAI Gym implementations for all tasks. For policy optimization we used the Proximal Policy Optimization (PPO) [7] implementation from OpenAI Baselines [4].

4.1 MuJoCo

We first examined the performance of T-REX on the HalfCheetah, Hopper, and Ant locomotion tasks. In all three tasks, the goal of the agent is to move forward as fast as possible without falling to the ground. To generate demonstrations, we trained a PPO agent for 500 training steps (64,000 simulation steps) and saved its policy after every 5 training steps. This provides us with different policies of varying quality. For each checkpoint we generated a trajectory of length 1,000. We then ranked the trajectories based on the ground truth returns. We divided the trajectories into three stages, consisting of overlapping training sets. We used 3 stages for HalfCheetah (the first 9, 12, and 24 trajectories) and Hopper (the first 9, 12, and 18 trajectories). For Ant we used two stages with the first 12 and 40 trajectories, respectively. We trained the reward network using 5,000 random pairs of partial trajectories of length 50, with preference labels based on the trajectory rankings, not the ground-truth return of the partial trajectories. We represented the reward function using an ensemble of five deep neural networks, trained separately with different random pairs. Each network has 3 fully connected layers of 256 units with ReLU nonlinearities. It was trained using the Adam optimizer with a learning rate of 1e-4 and a minibatch size of 64 for 10,000 time steps.

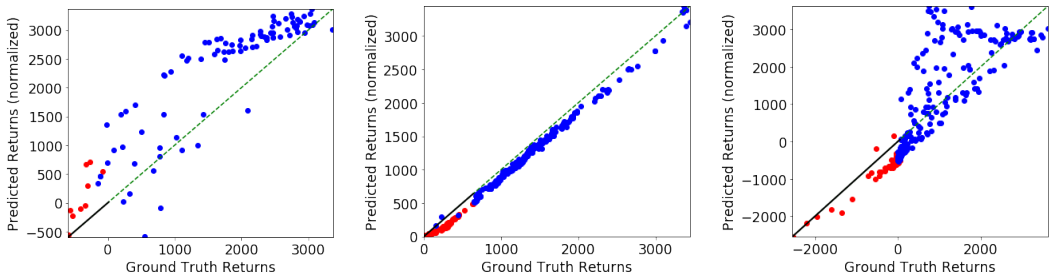
To evaluate the quality of our learned reward, we then trained a policy to maximize the inferred reward function via PPO. The agent receives the average of the ensemble as the reward, plus the control penalty used in OpenAI Gym. This control penalty represents a standard safety prior over reward functions for robotics tasks, namely to minimize joint torques. We found that optimizing a policy based solely on this control penalty does not lead to forward locomotion, thus learning a reward function from demonstrations is still necessary.

Learned Policy Performance We measured the performance of the policy learned by T-REX under the ground-truth reward function. We compared against Behavior Cloning from Observations (BCO) [8], a state-of-the-art learning from observation method, and Generative Adversarial Imitation Learning (GAIL) [5], a state-of-the-art IRL method. GAIL requires action labels, whereas T-REX and BCO learn from state observations only. We compared against three different levels of suboptimality (Stage 1, 2, and 3), corresponding to increasingly better demonstrations. The results are shown in Table 1. The policies learned by T-REX perform significantly better than the provided suboptimal trajectories in all the stages of HalfCheetah and Hopper. This provides evidence that T-REX can discover reward functions that extrapolate beyond the performance of the demonstrator, while behavioral cloning fails to perform better than the average demonstration performance in all tasks.

Reward Extrapolation We next investigated the ability of T-REX to accurately extrapolate beyond the demonstrator. To do so, we compared ground-truth return and T-REX-inferred return across trajectories from a range of performance qualities, including trajectories much better than the best demonstration given to T-REX. The extrapolation of the reward function learned by T-REX is shown in Figure 1. The plots in Figure 1 give insight into the performance of T-REX. When T-REX learns a reward function that has a strong positive correlation between the ground-truth reward function and the inferred reward function, then it is able to surpass the performance of the suboptimal demonstrations. However, in Ant the correlation is not as strong, resulting in worse-than-demonstrator performance in Stage 2.

Table 1: The results on three robotic locomotion tasks when given suboptimal demonstrations. For each stage and task, the best performance given suboptimal demonstrations is shown on the top row, and the best achievable performance using RL on the ground-truth reward is shown on the bottom row. The mean and standard deviation are based on 25 trials (obtained by running PPO five times and for each run of PPO performing five policy rollouts).

	HalfCheetah			Hopper			Ant	
	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2
Best Demo Performance	12.52 (1.04)	44.98 (0.60)	89.87 (8.15)	3.70 (0.01)	5.40 (0.12)	7.95 (1.64)	1.56 (1.28)	**54.64** (22.09)
T-REX	46.90 (1.89)	**61.56** (10.96)	143.40 (3.84)	**15.13** (3.21)	10.10 (1.68)	**15.80** (0.37)	4.93 (2.86)	7.34 (2.50)
BCO	7.71 (8.35)	23.59 (8.33)	57.13 (19.14)	3.52 (0.14)	4.41 (1.45)	4.58 (1.07)	1.06 (1.79)	26.56 (12.96)
GAIL	7.39 (4.12)	8.42 (3.43)	26.28 (12.73)	8.09 (3.25)	10.99 (2.35)	12.63 (3.66)	0.95 (2.06)	5.84 (4.08)
Best w/ GT Reward	199.11 (9.08)			15.94 (1.47)			182.23 (8.98)	



(a) HalfCheetah (b) Hopper (c) Ant

Figure 1: Extrapolation plots for T-REX for MuJoCo Stage 1 demonstrations. Red points correspond to demonstrations and blue points correspond to trajectories generated from PPO checkpoints not given as demonstrations. All predicted returns are normalized to have the same scale as the ground-truth returns.

4.2 Atari

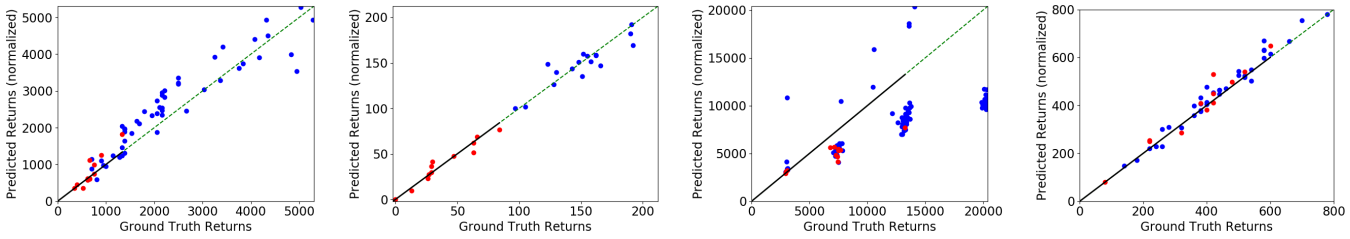
In the next set of experiments, we evaluated T-REX on eight Atari games. For each game we generated 12 full-episode trajectories using PPO policies checkpointed every 50 training updates for all games except for Seaquest where we used every 5th training update due to the faster learning in this game. We used an architecture for reward learning similar to the one proposed by Ibarz et al. (2018), with four convolutional layers with sizes 7x7, 5x5, 3x3, and 3x3, with strides 3, 2, 1, and 1. Each convolutional layer used 16 filters and LeakyReLU non-linearities. We then used a fully connected layer with 64 hidden units and a single scalar output. We fed in stacks of 4 frames with values normalized between 0 and 1. We additionally mask the Atari game score and number of lives. We optimized the reward function using Adam with a learning rate of 5e-5. For all games except for Enduro we train the reward network using partial trajectories between 50 and 100 observations long. For Enduro, we used randomly subsampled full trajectories, which resulted in much better performance. Given the learned reward function, we optimized a policy by training a PPO agent on the learned reward function for 50 million frames. We normalized the learned reward function by feeding the output of $\hat{r}_\theta(s)$ through a sigmoid function before passing it to the PPO algorithm. We ran experiments on the eight different Atari games shown in Table 2.

Learned Policy Performance The average performance of T-REX under the ground-truth reward function and the best and average performance of the demonstrator are shown in Table 2. We also compare against Behavioral Cloning from Observation (BCO) [8] and Generative Adversarial Imitation Learning (GAIL) [5]. Table 2 shows that T-REX outperforms BCO and GAIL in 7 out of 8 games. More importantly, T-REX outperforms the best demonstration in 7 out of 8 games.

Reward Extrapolation We also examined the extrapolation of the reward function learned using T-REX on several games. Results are shown in Figure 2. T-REX achieves accurate extrapolation between normalized predicted and ground-truth returns for Beam Rider, Enduro, and Seaquest—three games where we are able to outperform the best demonstration. The extrapolation plot for Hero has less correlation between ground-truth and predicted rewards which is likely the cause of the poor performance of T-REX on this game.

Table 2: Comparison of T-REX with a state-of-the-art behavioral cloning algorithm (BCO) [8] and state-of-the-art IRL algorithm (GAIL) [5]. Performance is evaluated on the ground-truth reward. T-REX achieves better-than-demonstrator performance on 7 out of 8 games and surpasses the BCO and GAIL baselines on 7 out of 8 games. Results are the best average performance over 3 random seeds with 30 trials per seed.

	Ranked Demonstrations		LfD Algorithm Performance		
Game	Best	Average	T-REX	BCO	GAIL
Beam Rider	1,332	686.0	**3,335.7**	568	355.5
Breakout	32	14.5	**221.3**	13	0.28
Enduro	84	39.8	**586.8**	8	0.28
Hero	**13,235**	6,742.0	0	2,167	0
Pong	-6	-15.6	**-2.0**	-21	-21
Q*bert	800	627	**32,345.8**	150	0
Seaquest	600	373.3	**747.3**	0	0
Space Invaders	600	332.9	**1,032.5**	88	370.2



(a) Beam Rider (b) Enduro (c) Hero (d) Seaquest

Figure 2: Ground-truth returns over demonstrations compared with the predicted returns using T-REX (normalized to be in the same range as the ground-truth returns). Red points represent suboptimal demonstration trajectories used to train the reward function. Blue data points represent trajectories not given as demonstrations. The solid line represents the performance range of the demonstrator. Games with more accurate extrapolation have better performance (see Table 2).

5 Conclusion and Future Work

In this paper, we introduced T-REX, a reward learning technique for high-dimensional tasks that can learn to extrapolate intent from suboptimal ranked demonstrations. To the best of our knowledge, this is the first IRL algorithm that is able to significantly outperform the demonstrator and that scales to high-dimensional Atari games. In the future, we plan to study the robustness of T-REX when given noisy rankings over demonstrations and to apply T-REX to actual human demonstrations.

References

[1] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.

[2] Daniel S Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. *arXiv preprint arXiv:1904.06387*, 2019.

[3] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[4] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[5] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[6] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. In *Advances in Neural Information Processing Systems*, pages 8022–8034, 2018.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[8] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *International Joint Conference on Artificial intelligence*, 2018.

A Top-down, Bottom-up Attention Model for Reinforcement Learning

Mehraveh Salehi
Department of Electrical Engineering
Yale University
mehraveh.salehi@yale.edu

Eser Aygun
Google DeepMind
eser@google.com

Shibl Murad
Google DeepMind
shibl@google.com

Doina Precup
Google DeepMind
doinap@google.com

Abstract

Reinforcement Learning (RL) agents typically have to process massive amounts of sensory data in order to execute a specific task. However, a large portion of the sensory input may not be directly related to the task at hand. Here, inspired by the human brain's attention system, we develop a novel augmented attention mechanism for RL agents, which enables them to adaptively select the most relevant information from the input. In order to evaluate the proposed algorithms, we use an attention-demanding grid-world environment and compare our model's performance against two other attentive agents and one naive agent. We demonstrate that our proposed augmented attention model outperforms other agents both in terms of scalability and ability to perform transfer learning.

Keywords: Attention, Reinforcement Learning, Neuroscience

Acknowledgements

Of the many who assisted in this work, we are especially thankful to Sasha Vezhnevets and Daniel Toyama.

1 Introduction

RL agents typically receive massive amounts of data from the environment. Processing these data can be expensive, both computationally and in terms of memory, yet the amount of relevant information about the task at hand may be small. Attention mechanisms have been developed to help RL agents adaptively select the most relevant information from the input and discard non-relevant data, without jeopardizing task performance [1, 7, 9]. However, it is often challenging to decide what makes a piece of information relevant to the task.

The majority of previous work on attention models has focused on supervised learning settings, where the relevance of information can be grounded in the model accuracy, measured using output labels. However, in RL settings, a piece of information which is not relevant at the current time step may be relevant later as the agent progresses through its trajectory as well as through the learning process. Therefore, a fixed strategy for attention might not be applicable. Furthermore, RL agents often deal with complex tasks, each including multiple sub-tasks. In such settings, there is a need for an adaptive attention mechanisms that are sensitive to the requirements of the task.

Recent works in RL have addressed the requirements of an adaptive attention in RL domain by designing an end-to-end model, where the attention is fully learned by the agent in parallel to the task[6, 5]. These approaches have shown great success in boosting the model performance, yet they require processing the entire environment before learning the attention mechanism. Despite their performance boost, their scalability to large environments and their generalizability to different settings is not clear. Here, we take inspiration from the human brain attention system and develop an augmented attention mechanism which both boosts performance and reduces computational cost.

2 Background and Motivation

The human brain attention system is complex and notoriously hard to characterize. Yet, one of the most influential models of attention describes it as a dual-network approach with two neural mechanisms [3]: (i) A top-down mechanism for orienting attention in a goal-driven or task-oriented fashion, and (ii) a bottom-up mechanism for directing attention in a stimulus-driven or observation-oriented fashion. For example, when we are reading a book, our brain executes the top-down attention as we are *actively* engaged in the reading task; as soon as our phone starts ringing, our bottom-up attention circuitry gets involved to adapt to the requirements coming from the environment, which may or may not be relevant to the task.

Inspired by this taxonomy of attention, we developed an augmented attention mechanism which includes both the top-down and the bottom-up attention systems. We evaluated our agent in an attention-demanding environment, designed such that it requires both task-oriented and observation-oriented attentions. We tested our agent's performance, scalability, and transferability in comparison to a naive agent (without any attention mechanism), an agent with top-down attention alone, and an agent with bottom-up attention alone. We will now report on our findings.

3 Methods

3.1 Environment Design

As the testing framework, we developed a grid world environment which includes 1 or more players, 1 or more obstacles, and 1 goal. At every time step, the agent has to decide which player to move and in what direction (left, right, up, and down). Next, the obstacles move with some probability in a random direction (left, right, up, and down), and if any of the players are hit by the obstacles, the game is over. The rewarding system is designed to promote both survival and catching the goal, so the agent collects +1 reward for every time step of survival and +500 reward if any of the players catches the goal.

3.2 Agent Design

The proposed model consists of two independent attention systems: (i) a top-down attention, which is learned in an end-to-end fashion parallel to the task, and (ii) a bottom-up attention, which is hard-wired in the model. At every time step t, the agent decides the attention parameters (Θ_t) and the action that it wants to perform (a_t), such that it maximizes the total expected return resulting from its interaction with the environment. We used Q-Learning [10] as our learning strategy. Below, we first describe each attention system separately, and then discuss how the two systems were combined to create the agent with the augmented attention. Note that although we use this environment for concreteness, our approahc is general and could be applied to other RL tasks with visual inputs.

3.2.1 Top-down attention

The top-down attention system (Figure 1a) is modeled as a sequential decision process, following the work by Mnih *et al.* [6] and Gregor *et al.* [5]. At every time-step, the agent observes the environment only partially through the lens of the attention kernel. This partial observation, called glimpse, is fed into a convolutional long short-term memory (ConvLSTM) unit. The output of the ConvLSTM is used to learn both the attention parameters Θ_t and the environment action a_t. The output is fed into a series of convolutional and linear layers and Q-values are generated for every state-action pair.

The attention model is adapted from DRAW [5] and contains an array of 2-dimensional Gaussian kernels with 5 parameters: the coordinates of the attention center (x, y), the distance between the kernels (δ), the isotropic variance (σ^2) and a scalar intensity (γ). These parameters are learned in an end-to-end fashion via back-propagation. The Gaussian attention kernel is multiplied by the current observation to generate the partial observation, or glimpse, for the agent.

3.2.2 Bottom-up attention

The bottom-up attention model (Figure 1b) is designed to only depend on the direct observation, and is hard wired to only include very limited information from the environment that is immediately related to the agent's survival. At every time step, this model crops the immediate neighborhood around each player and feeds it into a series of 1D convolutional layers. Similarly to the top-down approach, the entire RNN model is learned via backpropagation.

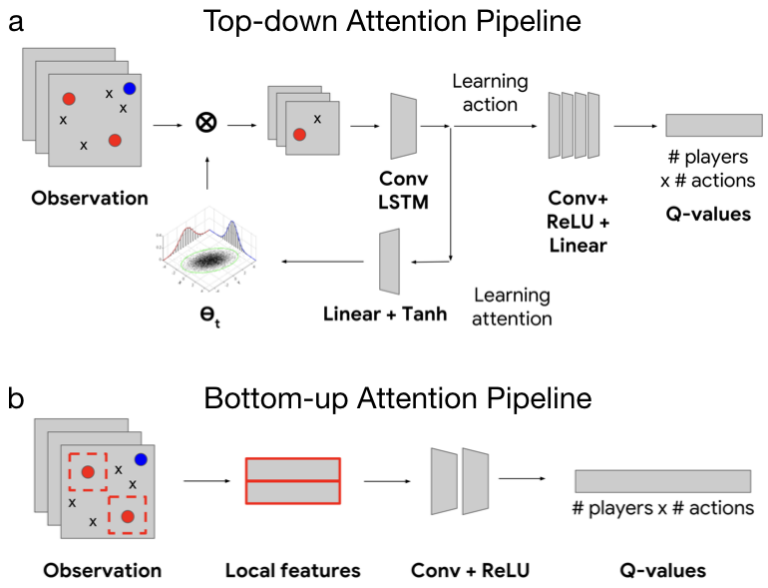


Figure 1: A visual illustration of the top-down and bottom-up attention systems

3.2.3 Augmented attention

The augmented attention model (Figure 2) combines the above attention systems using a linear layer. The output of each attention system is flattened and fed into a linear model, which then generates the Q-values. The entire model is learned end-to-end using back-propagation. Note that the top-down attention receives the output of the ConvLSTM model as the input, whereas the bottom-up attention receives input directly from the observation. This juxtaposition mimics our understanding of the human brain's top-down and bottom-up attention systems [8].

4 Experimental Results

We evaluated our augmented attention agent using two criteria: (i) scalability, and (ii) transferability. In all cases, we compared our agent with a naive Q-learning agent without attention, an agent using only the top-down attention system, and an agent using only the bottom-up attention system.

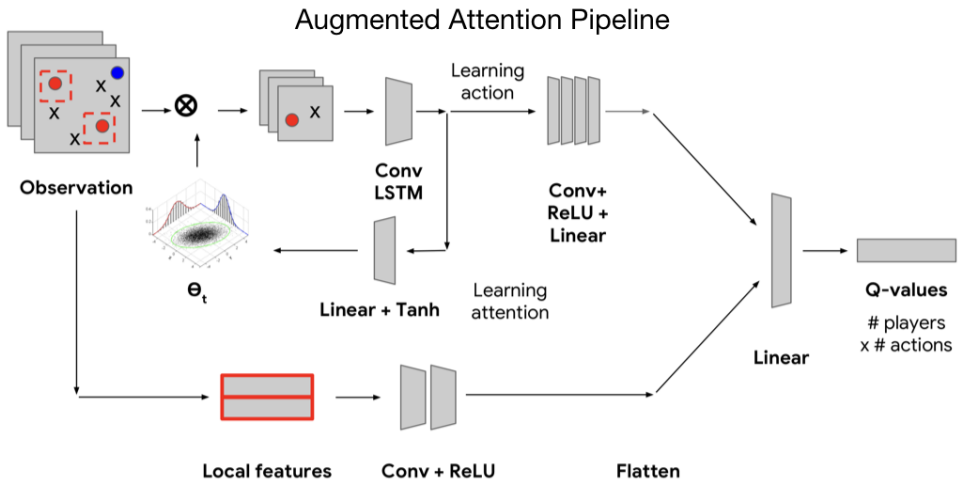


Figure 2: A visual illustration of the augmented attention system

4.1 Scalability

We assessed the scalability of all four agents by testing how their performance changes as we increase the world size and the number of players in the environment. We increased the world size from 10×10 with 5 obstacles to 20×20 with 20 obstacles. We also increased the complexity of the game by increasing the number of players from 1 to 5. Figure 3 displays the mean episode return for all four agents. We observe that the augmented attention scales better than all the other agents.

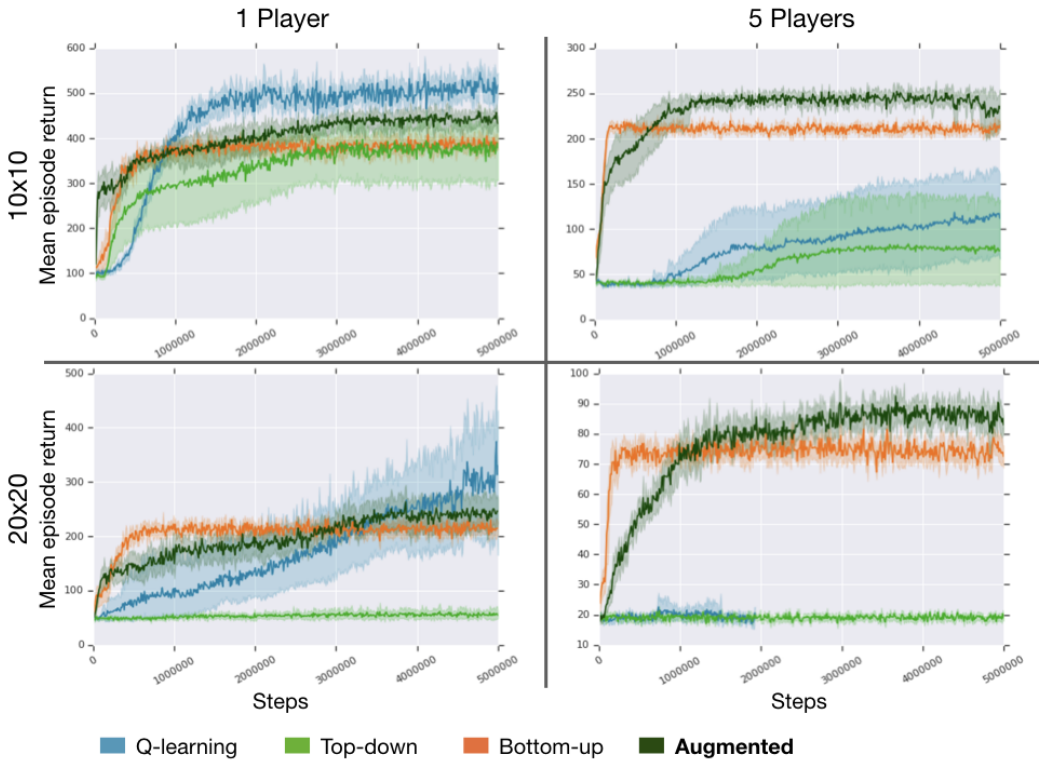


Figure 3: Evaluation of the model's scalability to larger world sizes and more players

4.2 Transferability

We assessed the transferability of the model using zero-shot transfer learning. We trained the agent in a 10×10 world with 5 obstacles and 2 players and tested the agent in a 50×50 world with 100 obstacles and 2 players. The naive Q-learning agent could not transfer because of its design limitations. The mean return after 10,000 time steps is reported in Table 1. We observe that the augmented attention model outperforms all the other agents in zero-shot transfer learning.

Attention Models			
	Top-down	Bottom-up	Augmented
Mean return	27.50	67.70	83.97

Table 1: Zero-shot learning evaluations

5 Conclusion and future work

In this work, inspired by the human brain attention mechanism, we designed an augmented attention system for an RL agent, which includes two independent components: a top-down attention which manages the goal-driven behavior of the agent, and a bottom-up attention, which gets triggered in a stimulus-driven fashion. Using an attention-demanding environment, we demonstrated that a model with augmented attention outperforms the agent without attention and the agents with only one attention system implemented, both in terms of scalability and transfer learning abilities. In the future, we hope to extend the experimental evaluation of this approach to other tasks. To take our findings back to the field of Neuroscience, we plan to extend our analysis to explicitly compare the proposed model with the human brain attention mechanism by employing a range of cognitive attention tasks that are widely used in the human brain research, such as the Attention Network Test (ANT) [4] and the Continuous Performance Task (CPT) [2].

References

[1] Jinyoung Choi, Beom-Jin Lee, and Byoung-Tak Zhang. Multi-focus attention network for efficient deep reinforcement learning. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] C Keith Conners, MHS Staff, V Connelly, S Campbell, M MacLean, and J Barnes. Conners continuous performance test ii (cpt ii v. 5). *Multi-Health Syst Inc*, 29:175–96, 2000.

[3] Maurizio Corbetta and Gordon L Shulman. Control of goal-directed and stimulus-driven attention in the brain. *Nature reviews neuroscience*, 3(3):201, 2002.

[4] Jin Fan, Bruce D McCandliss, Tobias Sommer, Amir Raz, and Michael I Posner. Testing the efficiency and independence of attentional networks. *Journal of cognitive neuroscience*, 14(3):340–347, 2002.

[5] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471, Lille, France, 07–09 Jul 2015. PMLR.

[6] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[7] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2790–2799. JMLR.org, 2016.

[8] Yair Pinto, Andries R van der Leij, Ilja G Sligte, Victor AF Lamme, and H Steven Scholte. Bottom-up and top-down attention are independent. *Journal of vision*, 13(3):16–16, 2013.

[9] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, 2015.

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Scalable methods for computing state similarity in deterministic Markov Decision Processes

Pablo Samuel Castro
Google Brain
psc@google.com

Abstract

Markov Decision Processes (MDPs) are the standard formalism for expressing sequential decision problems, typically in the context of planning or reinforcement learning (RL). One of the central components of this formalism is the notion of a set of states S. Each state in S is meant to encode sufficient information about the environment such that an agent can learn how to behave in a (mostly) consistent manner. There is no canonical way of defining the set of states for a problem. Indeed, improperly designed state spaces can have drastic effects on the learning algorithm. A stronger notion of state identity is needed that goes beyond the labeling of states and which is able to capture behavioral indistinguishability. We explore notions of behavioral similarity via state metrics and in particular those which assign a distance of 0 to states that are behaviorally indistinguishable. Our work builds on bisimulation metrics (Ferns et al., 2004) which provide us with theoretical properties such as guaranteeing states that are close to each other (with respect to the metric) will have similar optimal value functions. These metrics are unfortunately expensive to compute and require fully enumerating the states, which renders them impractical for problems with large (or continuous) state spaces.

We address this impracticality in the deterministic setting in two ways. The first is by providing a new sampling-based online algorithm for exact computation of the metric with convergence guarantees. The second is by providing a learning algorithm for approximating the metric using deep nets, enabling approximation even for continuous state MDPs. We provide empirical evidence of the efficacy of both.

The methods presented in this paper enable the use of these theoretically-grounded metrics in large planning and learning problems. Possible applications include state aggregation, policy transfer, automatic construction of temporally extended actions, and representation learning.

Keywords: markov decision processes reinforcement learning planning

Acknowledgements

The author would like to thank Marc G. Bellemare, Gheorghe Comanici, Carles Gelada, Doina Precup, and everyone else in the Google Brain team in Montreal for their helpful suggestions and advice in preparing this work. Additionally, the author would like to thank the reviewers for their helpful comments.

1 Introduction

Markov Decision Processes (MDPs) are the standard formalism for expressing sequential decision problems, typically in the context of planning or reinforcement learning (RL). One of the central components of this formalism is a set of *states* $\mathcal{S}$. Each state in $\mathcal{S}$ is meant to encode sufficient information about the environment such that an agent can learn how to behave in a (mostly) *consistent* manner. Figure 1 illustrates a simple MDP where each cell represents a state.

There is no canonical way of defining the set of states for a problem. Indeed, improperly designed state spaces can have drastic effects on the learning algorithm. Consider the grid MDP in the bottom of Figure 1, where an agent must learn how to navigate to the green cells, and imagine we create an exact replica of the MDP such that the agent randomly transitions between the two layers for each move. By doing so we have doubled the number of states and the complexity of the problem. However, from a planning perspective the two copies of each state should be indistinguishable. A stronger notion of *state identity* is needed that goes beyond the labeling of states and which is able to capture *behavioral indistinguishability* (Castro (2011) provides a thorough investigation of behavioral equivalence in MDPs).

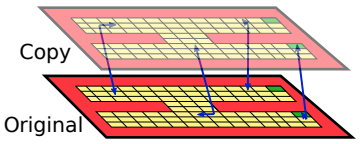


Figure 1: A grid MDP (bottom) with a copy of itself (top). The goal of an agent is to find the shortest path to the green cells. Blue arrows indicate transitions that can switch between the two copies.

We explore notions of behavioral similarity via state metrics $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, and in particular those which assign a distance of 0 to states that are behaviorally indistinguishable.

Our work builds on bisimulation metrics (Ferns et al., 2004) which provide us with theoretical properties such as guaranteeing states that are close to each other (with respect to the metric) will have similar optimal value functions. These metrics are unfortunately expensive to compute (Chen et al. (2012) provides a theoretical analysis) and require fully enumerating the states, which renders them incompatible with large state spaces.

2 Background

Definition 1. *A* **finite Markov Decision Process (MDP)** *is defined as a 5-tuple* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$*, where*

- $\mathcal{S}$ *is a finite set of states*
- $\mathcal{A}$ *is a finite set of actions*
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to Dist(\mathcal{S})$ *is the next state transition function*[1]
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is the reward function (assumed to be bounded by* R_{max}*)*
- $\gamma \in [0, 1)$ *is a discount factor*

Each *policy* $\pi : \mathcal{S} \to Dist(\mathcal{A})$ induces a corresponding state-value function $V^\pi : \mathcal{S} \to \mathbb{R}$, known as the Bellman equation (Bellman, 1957):

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(s)} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^\pi(s') \right]$$

In the control setting, we are typically interested in finding the optimal value function V^*:

$$V^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}(s,a)} V^*(s') \right]$$

Bisimulation relations, originally introduced in the field of concurrency theory, was adapted for MDPs by Givan et al. (2003). They capture a strong form of behavioral equivalence: if two states $s, t \in \mathcal{S}$ are bisimilar, then $V^*(s) = V^*(t)$.

Definition 2. *Given an MDP* $\mathcal{M}$*, an equivalence relation* $E \subseteq \mathcal{S} \times \mathcal{S}$ *is a* **bisimulation relation** *if whenever* $(s, t) \in E$ *the following properties hold, where* $\mathcal{S}_E$ *is the state space* $\mathcal{S}$ *partitioned into equivalence classes defined by* E*:*

1. $\forall a \in \mathcal{A}. \quad \mathcal{R}(s, a) = \mathcal{R}(t, a)$
2. $\forall a \in \mathcal{A}. \forall c \in \mathcal{S}_E. \mathcal{P}(s, a)(c) = \mathcal{P}(t, a)(c)$*, where* $\mathcal{P}(x, y)(c) = \sum_{z \in c} \mathcal{P}(x, y)(z)$

Two states $s, t \in \mathcal{S}$ *are* **bisimilar** *if there exists a bisimulation relation* E *such that* $(s, t) \in E$*. Note that there can be a number of equivalence relations satisfying these properties. The smallest is the identity relation, which is vacuously a bisimulation relation. We are interested in the largest bisimulation relation, which we will denote as* $\sim$*.*

[1] $Dist(X)$ denotes a probability distribution over the set X.

Ferns et al. (2004) generalized the notion of MDP bisimulation relations to metrics. Let $\mathbb{M}$ be the set of all metrics on $\mathcal{S}$. A pseudometric[2] $d \in \mathbb{M}$ induces an equivalence relation R_d by equating all states with zero distance. Ferns et al. (2004) defines a pseudometric $d \in \mathbb{M}$ as a **bisimulation metric** if R_d is $\sim$.

The following theorem introduces the operator $\mathcal{F}$ which can be used to iteratively compute a bisimulation metric, where $\mathcal{W}(d)$ is the Wasserstein metric between two probability distributions under the state metric d (Villani, 2008).

Theorem 1. *(Ferns et al., 2004): Define $\mathcal{F} : \mathbb{M} \to \mathbb{M}$ by*

$$\mathcal{F}(d)(s,t) = \max_{a \in \mathcal{A}} \left(|\mathcal{R}(s,a) - \mathcal{R}(t,a)| + \gamma \mathcal{W}(d)(\mathcal{P}(s,a), \mathcal{P}(t,a))\right)$$

Then $\mathcal{F}$ has a least-fixed point, $d_\sim$, and $d_\sim$ is a bisimulation metric.

3 Bisimulation metrics for deterministic MDPs

Although the study of bisimulation metrics has largely focused on MDPs with stochastic transitions, there are many problems of interest with deterministic transitions. By focusing on these types of problems we are able to design a new set of algorithms that can handle large or continuous state spaces. We believe this provides a strong foundation for future research extending these ideas to stochastic environments.

Definition 3. *A* deterministic *MDP $\mathcal{M}$ is one where all next-state transitions $\mathcal{P}(\cdot,\cdot)$ are deterministic. For convenience, we denote this unique next state as $\mathcal{N}(s,a)$.*

Lemma 1. *Given a deterministic MDP $\mathcal{M}$, for any two states $s,t \in \mathcal{S}$, action $a \in \mathcal{A}$, and pseudometric $d \in \mathbb{M}$,*

$$\mathcal{W}(d)(\mathcal{P}(s,a), \mathcal{P}(t,a)) = d(\mathcal{N}(s,a), \mathcal{N}(t,a))$$

By Lemma 1 we can rewrite the equation in Theorem 1 as: $\mathcal{F}(d)(s,t) = \max_{a \in \mathcal{A}} \left(|\mathcal{R}(s,a) - \mathcal{R}(t,a)| + \gamma d(\mathcal{N}(s,a), \mathcal{N}(t,a))\right)$. Note the close resemblance to the Bellman equation. There is in fact a strong connection between the two: Ferns & Precup (2014) proved that $d_\sim$ is the optimal value function of an optimal coupling of two copies of the original MDP.

4 Computing bisimulation metrics with sampled trajectories

The update operator $\mathcal{F}$ is generally applied in a dynamic-programming fashion: all state-pairs are updated in each iteration by considering all possible actions. However, requiring access to all state-pairs and actions in each iteration is often not possible, especially when data is concurrently being collected by an agent interacting with an environment. In this section we present an algorithm for computing the bisimulation metric via access to *trajectory samples*. Specifically, assume we are able to sample pairs of transitions $\{\langle s,a,\mathcal{R}(s,a),\mathcal{N}(s,a)\rangle, \langle t,a,\mathcal{R}(t,a),\mathcal{N}(t,a)\rangle\}$ from an underlying distribution $\mathcal{D}$ (note the action is the same for both). This can be, for instance, a uniform distribution over all transitions in a replay memory (Mnih et al., 2015) or some other sampling procedure. Let $\mathcal{T}$ be the set of all pairs of valid transitions; for legibility we will use the shorthand $\tau_{s,t,a} \in \mathcal{T}$ to denote a pair of transitions from states $s,t \in \mathcal{S}$ under action $a \in \mathcal{A}$. We assume that $\mathcal{D}(\tau) > 0$ for all $\tau \in \mathcal{T}$.

Theorem 2. *Let $d_0 \equiv 0$ be the everywhere-zero metric. At step n, let $\tau_{s_n,t_n,a_n} \in \mathcal{T}$ be a sample from $\mathcal{D}$ and define d_n as:*

- $d_n(s_n,t_n) = \max\left(d_{n-1}(s_n,t_n), |\mathcal{R}(s_n,a_n) - \mathcal{R}(t_n,a_n)| + \gamma d_{n-1}(\mathcal{N}(s_n,a_n), \mathcal{N}(t_n,a_n))\right)$
- $d_n(s,t) = d_{n-1}(s,t). \qquad \forall s \neq s_n, t \neq t_n$

Then $\lim_{n\to\infty} d_n = d_\sim$ almost surely.

In Figure 2 we demonstrate the efficacy of our iterative procedure on grid world domains of varying sizes. The results suggest that even in cases where we do have access to all state-pairs and actions at each iteration (as is required by the traditional dynamic-programming approach), using the sampling method we introduce here can yield a significant computational advantage.

$\lvert\mathcal{S}\rvert$	DP Method	Sampling method
9	0.477 ± 0.002	0.027 ± 0.0002
25	4.244 ± 0.002	0.357 ± 0.0005
100	63.705 ± 0.011	16.058 ± 0.0046
400	915.609 ± 0.1062	252.547 ± 0.0407
900	5128.54 ± 0.5255	2164.077 ± 0.6043

Figure 2: Comparison of running time (in seconds) between the standard DP method and our proposed sampling method.

5 Learning an approximation

The previous section addresses the case when states are only accessible via sampling of the underlying state space, but still requires full enumerability of the states. In this section we present a method for dealing with continuous state spaces by means of three key components (Sections 5.1, 5.2, and 5.3).

[2] A pseudometric is a metric d where $\forall s,t \in \mathcal{S}. s = t \implies d(s,t) = 0$, but not the converse.

5.1 Approximating bisimulation metrics with deep neural networks

We make use of deep neural networks to approximate the bisimulation distance between any two states in an MDP. Let $\phi : \mathcal{S} \to \mathbb{R}^k$ be a k-dimensional representation of the state space and let $\psi_\theta : \mathbb{R}^{2k} \to \mathbb{R}$ be a deep neural network parameterized by θ that receives a concatenation of two state representations such that $\psi_\theta([\phi(s), \phi(t)]) \approx d_\sim(s,t)$. Figure 3 illustrates one such network with a single hidden layer of dimension h.

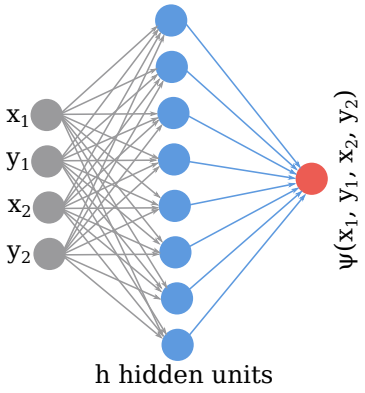


Figure 3: The network topology used for learning ψ as an approximant to $d_\sim$

5.2 A differentiable loss

Following the practice introduced by Mnih et al. (2015) we make use of online parameters θ and target parameters θ^-, where the online parameters are updated at each iteration while the target parameters are updated every C iterations. Given a pair of states $s \neq t$ and action $a \in \mathcal{A}$, at iteration i we define our target objective $\mathbf{T}_{\theta_i^-}(s,t,a)$ as:

$$\max\left(|\mathcal{R}(s,a) - \mathcal{R}(t,a)| + \gamma\psi_{\theta_i^-}([\phi(\mathcal{N}(s,a)), \phi(\mathcal{N}(t,a))]), \psi_{\theta_i^-}([\phi(s),\phi(t)])\right)$$

and equal to 0 whenever $s = t$.

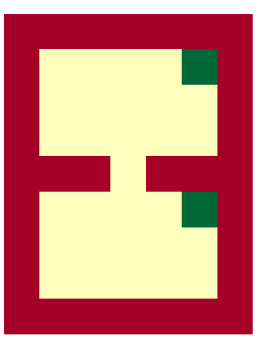
Our loss is then: $\mathcal{L}_{s,t,a} = \mathbb{E}_\mathcal{D}\left(\mathbf{T}_{\theta_i^-}(s,t,a) - \psi_{\theta_i}([\phi(s),\phi(t)])\right)^2$.

5.3 Learning with mini-batches

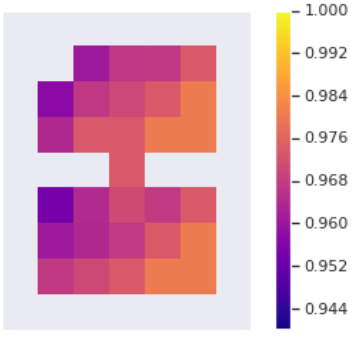
We proceed to specify a set of matrix operations for computing them on a *batch* of states. This allows us to efficiently train this approximant using specialized hardware like GPUs. At each step we assume access to a batch of b samples of states $\mathbf{S}$, actions $\mathbf{A}$, rewards $\mathbf{R}$, and next states $\mathbf{N}$, where $\mathbf{S}_i = \phi(s_i)$, $\mathbf{R}_i = \mathcal{R}(s_i, a_i)$, etc. Letting $[X, Y]$ stand for the concatenation of two vectors X and Y, from $\mathbf{S}$ we construct a new square matrix $\mathbf{S}^2$ of dimension $b \times b$ such that $\mathbf{S}^2_{i,j} = [\phi(s_i), \phi(s_j)]$. Each element in this matrix is a vector of dimension $2k$. We then reshape this matrix to be a "single-column" tensor of length b^2. We can perform a similar exercise on the reward and next-state batches. Finally, we define a mask $\mathbf{W}$ which enforces that we only consider pairs of samples that have matching actions: $\mathbf{W}_{i,j} = [a_i == a_j]$.

In batch-form, the target defined above becomes: $\mathbf{T} = (1 - \mathbf{I}) * \max\left(\mathbf{R}^2 + \gamma\beta\psi_{\theta_i^-}(\mathbf{N}^2), \quad \beta\psi_{\theta_i^-}(\mathbf{S}^2)\right)$.

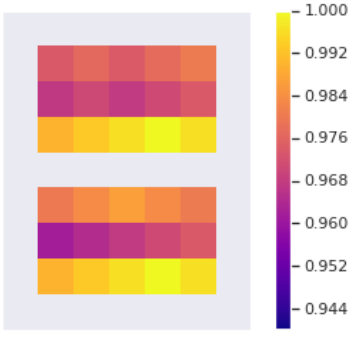
where $\psi(\mathbf{X})$ indicates applying ψ to a matrix $\mathbf{X}$ elementwise. We multiply by $(1 - \mathbf{I})$ to zero out the diagonals, since those represent approximations to $d_\sim(s,s) \equiv 0$. The parameter β is a stability parameter that begins at 0 and is incremented every C iterations. Its purpose is to gradually "grow" the effective horizon of the bisimulation backup and maximization. This is necessary since the approximant ψ_θ can have some variance initially, depending on how θ is initialized; in particular, we cannot in general guarantee $\psi_{\theta_0} \equiv 0$, as is required by Theorem 2. Further, Jiang et al. (2015) demonstrate that using shorter horizons during planning can often be better than using the true horizon, especially when using a model estimated from data. Finally, our loss $\mathcal{L}_i$ at iteration i is defined as: $\mathcal{L}_i(\theta_i) = \mathbb{E}_\mathcal{D}\left[\mathbf{W} \otimes \left(\psi_{\theta_i}(\mathbf{S}^2) - \mathbf{T}\right)^2\right]$, where $\otimes$ stands for the Hadamard product. Note that, in general, the approximant ψ is not a metric: it can violate the identity of indiscernibles, symmetry, and subadditivity conditions.



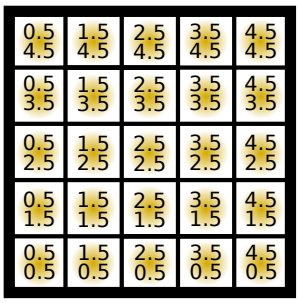
(a) GridWorld reward dynamics.



(b) Bisimulation distances between top left cell and the rest of the cells.

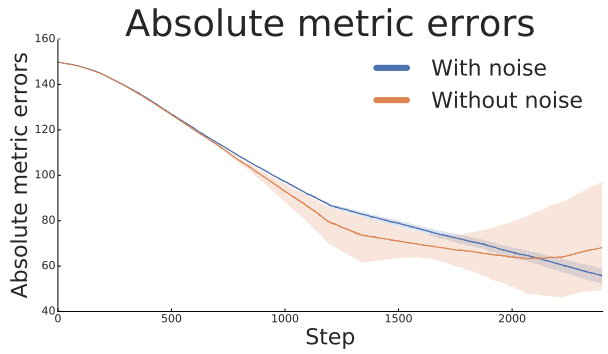


(c) Bisimulation distances between the hallway cell and the rest of the cells.

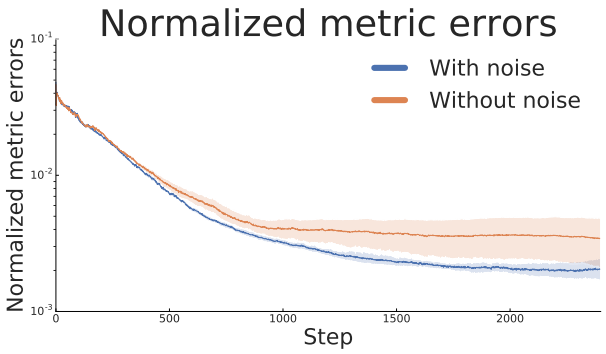


(d) 5x5 GridWorld with noise (in yellow). The top and bottom numbers in each cell are the x and y coords.

Figure 4: GridWorld dynamics and bisimulation distances from two rooms. The distances from the center cell highlight the symmetries in the environment.



(a) Absolute metric errors on the GridWorld.



(b) Normalized metric errors.

Figure 5: Metric errors for the learned metric as training progresses over 10 independent runs; the shaded areas represent the 95% confidence interval. $\gamma = 0.99$, $C = 500$, $b = 256$, $\beta = 0.9$, and $h = 729$; we used the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.01.

5.4 Empirical Evaluation

We evaluate our learning algorithm in the 31-state GridWorld environment illustrated in Figure 4a. There are 4 actions (up, down, left, right) with deterministic transitions, and where an action driving the agent towards a wall keeps the agent in the same cell. There is a single reward of $+1.0$ received upon entering either of the green cells, and a reward of -1.0 for taking an action towards a wall. We display the bisimulation distances relative to two cells in this environment in Figure 4b and Figure 4c. The distances highlight the symmetries in the environment, especially Figure 4c.

We represent each state by its coordinates $(x, y) \in \mathbb{R}^2$, as illustrated in Figure 4d. To estimate $d_\sim$ we use a network with an input layer of dimension 4, one fully connected layer of length h, and an output of length 1. The input is the concatenation of two state representations, normalized to be in $[-1, 1]$, while the output value is the estimate to $d_\sim$.

To evaluate the learning process, we measure $\|d_\sim - \psi\|_\infty$ (absolute metric errors) as well as $\|\frac{d_\sim}{\|d_\sim\|_2} - \frac{\psi}{\|\psi\|_2}\|_\infty$ (normalized metric errors) using the true underlying state space for which we know the value of $d_\sim$ (Figure 5a and Figure 5b).

In addition to training the network on the 31 states, we experimented with adding noise to the state representations. Specifically, when a state (x, y) is sampled, we add Gaussian noise centered at $(0, 0)$ with stddev 0.1, and clipped to be in $[-0.3, 0.3]$, effectively converting the 31 state MDP into a continuous-state MDP (e.g. Figure 4d).

Because our estimate ψ is part of the maximization in the target objective, it can be difficult to prevent continuous growth of the metric approximants. This can be seen happening towards the end of training in the line without noise in Figure 5a.

Although the normalized errors suggest that the relative magnitudes of the distances remain stable, we found that doubling the value of C halfway through training helped mitigate this "overshooting" phenomenon.

As can be seen, there is little difference between learning the metric for the 31-state MDP versus learning it for the continuous MDP. In fact, training under the continuous MDP seems to add stability and improve performance. This suggests that the learning process suffers more from the overshooting phenomenon when there are few data points to train on: the continuum of data points in the continuous setting are possibly helping regularize the network.

Finally, in Figure 6 we explore aggregating a set of states sampled from the continuous MDP. We sampled 100 independent samples for each underlying cell, computed the distances between each pair of sampled states, and then aggregated them by incrementally growing "clusters" of states while ensuring that all states in a cluster are within a certain distance of each other. As can be seen, our learned distance is able to capture many of the symmetries present in the environment: the orange cluster tends to gather near-goal states, the dark-brown and dark-blue clusters seem to gather states further away from goals, while the bright red states properly capture the unique "hallway" cell. This experiment highlights the potential for successfully approximating bisimulation metrics in continuous state MDPs, which can render continuous environments more manageable.

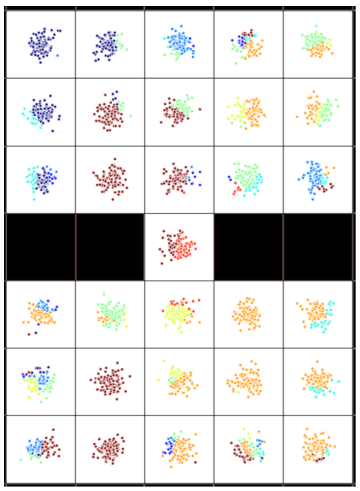


Figure 6: Aggregating samples drawn from a continuous MDP using the learned bisimulation metric approximant.

References

Bellman, R. *Dynamic Programming*. University Press, Princeton, NJ, USA, 1957.

Castro, P. S. *On planning, prediction and knowledge transfer in Fully and Partially Observable Markov Decision Processes*. PhD thesis, McGill University, 2011.

Chen, D., van Breugel, F., and Worrell, J. On the Complexity of Computing Probabilistic Bisimilarity. In *Foundations of Software Science and Computational Structures*, pp. 437–451, 2012.

Ferns, N. and Precup, D. Bisimulation Metrics are Optimal Value Functions. 2014.

Ferns, N., Panangaden, P., and Precup, D. Metrics for Finite Markov Decision Processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pp. 162–169, 2004.

Givan, R., Dean, T., and Greig, M. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147:163–223, July 2003.

Jiang, N., Kulesza, A., Singh, S., and Lewis, R. The Dependence of Effective Planning Horizon on Model Accuracy. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-15)*, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 2015.

Villani, C. *Optimal Transport*. Springer-Verlag Berlin Heidelberg, 2008.

Constrained Policy Improvement for Safe and Efficient Reinforcement Learning

Elad Sarafian
Bar-Ilan University, Israel
elad.sarafian@gmail.com

Aviv Tamar
Technion, Israel
aviv.tamar.mail@gmail.com

Sarit Kraus
Bar-Ilan University, Israel
sarit@cs.biu.ac.il

Abstract

We propose a policy improvement algorithm for Reinforcement Learning (RL) which is called Rerouted Behavior Improvement (RBI). RBI is designed to take into account the evaluation errors of the Q-function. Such errors are common in RL when learning the Q-value from finite past experience data. Greedy policies or even constrained policy optimization algorithms which ignore these errors may suffer from an improvement penalty (i.e. a negative policy improvement). To minimize the improvement penalty, the RBI idea is to attenuate rapid policy changes of low probability actions which were less frequently sampled. This approach is shown to avoid catastrophic performance degradation and reduce regret when learning from a batch of past experience. Through a two-armed bandit with Gaussian distributed rewards example, we show that it also increases data efficiency when the optimal action has a high variance. We evaluate RBI in two tasks in the Atari Learning Environment: (1) learning from observations of multiple behavior policies and (2) iterative RL. Our results demonstrate the advantage of RBI over greedy policies and other constrained policy optimization algorithms as a safe learning approach and as a general data efficient learning algorithm. A Github repository of our RBI implementation is found at https://github.com/eladsar/rbi/tree/rbi.

Keywords: Safe Reinforcement Learning, Constrained Policies Algorithms

Acknowledgements

This research has been partly supported by the Israel Innovation Authority and by the Ministry of Science and Technology, Israel.

1 Introduction

While Deep Reinforcement Learning (DRL) is the backbone of many of the recent Artificial Intelligence breakthroughs [14, 10], it suffers from several factors which inhibit deployment of RL systems to real-world tasks. Two of these elements are: (1) data efficiency and; (2) safety. DRL is notoriously data and time inefficient, requires up to billions of history states [6] or weeks of wall-clock time [5] to train to expert level. While it is partially due to the slow training process of deep neural networks, it is also due to inefficient, yet simple to implement, policy improvement routines. For example, a greedy policy improvement (with a fix exploration parameter) is known to have a higher regret than other methods such as Upper Confidence Bound (UCB) [2], but the latter is much more difficult to adjust to a deep learning framework [3]. This transformation from the countable state space of bandit and grid-world problems to the uncountable state-space in a DRL framework, calls for efficient improvement methods which fit into existing deep learning frameworks.

For some real-world problems, like autonomous cars [13], safety is a crucial factor. Random initialized policies and even a RL algorithm that may suffer from sudden catastrophic performance degradation are both unacceptable in such environments. While policy initialization may be solved with Learning from Demonstrations (LfD) algorithms [1], changing the policy in order to improve performance is still a risky task. Largely since the Q-value of a current policy can only be estimated from the past data. Therefore, for safe RL, it is desirable to design improvement algorithms that model the accuracy of the Q-value evaluation and can mitigate between fast improvement and a safety level [4, 17].

In this work, we propose a policy improvement method that addresses both the sample efficiency of the learning process and the problem of safe learning from incomplete past experience. We start by analyzing the improvement penalty of an arbitrary new policy $\pi(a|s)$ based on an estimated Q-function of a past behavior policy $\beta(a|s)$. We find that under a simplified model of learning the Q-values from i.i.d samples, the variance of a potential improvement penalty is proportional to $\frac{|\beta(a|s)-\pi(a|s)|^2}{\beta(a|s)}$. Therefore, we design a constraint, called reroute, that limits this term. We show that finding the optimal policy under the reroute constraint amounts to solving a simple linear program. Instead of optimizing this policy via a gradient descent optimization, we take a different approach and solve it in the non-parameterized space for every new state the actor encounters. In order, to learn the new improved policy with a parameterized Neural Network (NN), we store the calculated policy into a replay buffer and imitate the actor's policy with a KL regression.

While RBI is designed for safe learning from a batch of past experience, we show that it also increase data efficiency with respect to a greedy step and other constraints such as the Total Variation (TV) [7] and PPO [12]. In fact it is akin in practice to the forward KL constraint [18], however, unlike the KL constraint, it does not require different scaling for different reward signals and it is much more intuitive to design. We validate our findings both in simple environments such as a two-armed bandit problem with Gaussian distributed reward and also in a complex distributed actors framework when learning to play Atari.

2 Rerouted Behavior Improvement

Let us start by examining a single improvement step from a batch of past experience of a behavior policy. Define by β the behavior policy of a dataset $\mathcal{D}$ and by Q^β, and $\hat{Q}^\beta$ its true and approximated Q-functions. Theoretically, for an infinite dataset with infinite number of visitations in each state-action pair, one may calculate the optimal policy in an off-policy fashion [19]. However, practically, one should limit its policy improvement step over β when learning from a realistic finite dataset. To design a proper constraint, we analyze the statistics of the error of our evaluation of $\hat{Q}^\beta$. This leads to an important observation: the Q-value has a higher error for actions that were taken less frequently, thus, to avoid improvement penalty, we must restrict the ratio of the change in probability $\frac{\pi}{\beta}$. We will use this observation to craft the reroute constraint, and show that other well-known monotonic improvement methods (e.g. PPO and TRPO) overlooked this consideration, hence they do not guarantee improvement when learning from a finite experience.

2.1 Soft Policy Improvement

Before analyzing the error's statistics, we begin by considering a set of policies which improve β if our estimation of Q^β is exact. Out of this set we will pick our new policy π. Recall that the most naive and also common improvement method is taking a greedy step, i.e. deterministically acting with the highest Q-value action in each state. This is known by the policy improvement theorem [16], to improve the policy performance. The policy improvement theorem may be generalized to include a larger family of soft steps.

Lemma 2.1 (Soft Policy Improvement)**.** *Given a policy β, with value and advantage V^β, A^β, a policy π improves β, i.e. $V^\pi \geq V^\beta$ $\forall s$, if it satisfies $\sum_a \pi(a|s)A^\beta(s,a) \geq 0$ $\forall s$ with at least one state with strict inequality. The term $\sum_a \pi(a|s)A^\beta(s,a)$ is called the improvement step.*[1]

[1]The proof adhere to the same steps of the greedy improvement proof in [16], thus it is omitted for brevity.

Essentially, every policy that increases the probability of taking positive advantage actions over the probability of taking negative advantage actions achieves improvement. Later, we will use the next Corollary to prove that RBI guarantees a positive improvement step.

Corollary 2.1.1 (Rank-Based Policy Improvement). *Let $(A_i)_{i=1}^{|\mathcal{A}|}$ be an ordered list of the β advantages in a state s, s.t. $A_{i+1} \geq A_i$, and let $c_i = \pi_i/\beta_i$. If for all states $(c_i)_{i=1}^{|\mathcal{A}|}$ is a monotonic non-decreasing sequence s.t. $c_{i+1} \geq c_i$, then π improves β.*

2.2 Standard Error of the Value Estimation

To provide a statistical argument for the expected error of the Q-function, consider learning $\hat{Q}^\beta$ with a tabular representation. The Q-function is the expected value of the random variable $z^\pi(s,a) = \sum_{k\geq 0} \gamma^k r_k | s, a, \pi$. Therefore, the Standard Error (SE) of an approximation $\hat{Q}^\beta(s,a)$ for the Q-value with N i.i.d. MC trajectories is

$$\sigma_{\varepsilon(s,a)} = \frac{\sigma_{z(s,a)}}{\sqrt{N_s \beta(a|s)}}, \tag{1}$$

where N_s is the number of visitations in state s in $\mathcal{D}$, s.t. $N = \beta(a|s)N_s$. Therefore, $\sigma_{\varepsilon(s,a)} \propto \frac{1}{\sqrt{\beta(a|s)}}$ and specifically for low frequency actions such estimation may suffer large SE.[2] Notice that we ignore recurrent visitations to the same state during the same episode and hence, the MC discounted sum of rewards are independent random variables.

2.3 Policy Improvement in the Presence of Value Estimation Errors

We now turn to the crucial question of what happens when one applies an improvement step with respect to an inaccurate estimation of the Q-function, i.e. $\hat{Q}^\beta$.

Lemma 2.2 (Improvement Penalty). *Let $\hat{Q}^\beta = \hat{V}^\beta + \hat{A}^\beta$ be an estimator of Q^β with an error $\varepsilon(s,a) = (Q^\beta - \hat{Q}^\beta)(s,a)$ and let π be a policy that satisfies lemma 2.1 with respect to $\hat{A}^\beta$. Then the following holds*

$$V^\pi(s) - V^\beta(s) \geq -\mathcal{E}(s) = -\sum_{s'\in\mathcal{S}} \rho^\pi(s'|s) \sum_{a\in\mathcal{A}} \varepsilon(s',a) \left(\beta(a|s') - \pi(a|s')\right), \tag{2}$$

where $\mathcal{E}(s)$ is called the improvement penalty and $\rho^\pi(s'|s) = \sum_{k\geq 0} \gamma^k P(s \xrightarrow{k} s'|\pi))$ is the unnormalized discounted state distribution induced by policy π.

Since $\varepsilon(s',a)$ is a random variable, it is worth to consider the variance of $\mathcal{E}(s)$. Define each element in the sum of Eq. (2) as $x(s',a;s) = \rho^\pi(s'|s)\varepsilon(s,a)(\beta(a|s') - \pi(a|s'))$. The variance of each element is therefore

$$\sigma^2_{x(s',a;s)} = (\rho^\pi(s'|s))^2 \sigma^2_{\varepsilon(s',a)} (\beta(a|s') - \pi(a|s'))^2 = \frac{(\rho^\pi(s'|s))^2 \sigma^2_{z(s',a)}}{N_{s'}} \frac{(\beta(a|s') - \pi(a|s'))^2}{\beta(a|s')}.$$

To see the the need for the reroute constraint, we can bound the total variance of the improvement penalty

$$\sum_{s',a} \sigma^2_{x(s',a;s)} \leq \sigma^2_{\mathcal{E}(s)} \leq \sum_{s',a,s'',a'} \sqrt{\sigma^2_{x(s',a;s)} \sigma^2_{x(s'',a';s)}},$$

where the upper bound is due to the Cauchy-Schwarz inequality, and the lower bound is since $\varepsilon(s,a)$ elements have a positive correlation (as reward trajectories overlap). Hence, it is evident that the improvement penalty can be extremely large when the term $\frac{|\beta-\pi|^2}{\beta}$ is unregulated and even a single mistake along the trajectory, caused by an unregulated element, might wreck the performance of the entire policy. However, by using the reroute constraint which tame each of these terms we can bound the variance of the improvement penalty.

While we analyzed the error for independent MC trajectories, a similar argument holds also for Temporal Difference (TD) learning [16]. [8] studied "bias-variance" terms in k-steps TD learning of the value function. Here we present their results for the Q-function error with TD updates. For any $0 < \delta < 1$, and a number t of iteration through the data for the TD calculation, the maximal error term abides

$$\varepsilon(s,a) \leq \max_{s,a} |\hat{Q}^\beta(s,a) - Q^\beta(s,a)| \leq \frac{1-\gamma^{kt}}{1-\gamma} \sqrt{\frac{3\log(k/\delta)}{N_s \beta(a|s)}} + \gamma^{kt}. \tag{3}$$

While the "bias", which is the second term in (3), depends on the number of iterations through the dataset, the "variance" which is the square root of the first term in (3) is proportional to $\frac{1}{\beta(a|s)N_s}$, therefore, bounding the ratio $\frac{|\beta-\pi|^2}{\beta}$ bounds the improvement penalty also for TD learning.

[2] Note that even for deterministic environments, a stochastic policy inevitably provides $\sigma_{z(s,a)} > 0$.

2.4 The Reroute Constraint

In order to confine the ratio $\frac{|\beta-\pi|^2}{\beta}$, we suggest limiting the improvement step to a set of policies based on the following constraint.

Definition 2.1 (Reroute Constraint)**.** Given a policy β, a policy π is a $reroute(c_{\min}, c_{\max})$ of β, if $\pi(a|s) = c(s,a)\beta(a|s)$ where $c(s,a) \in [c_{\min}, c_{\max}]$. Further, note that reroute is a subset of the TV constraint with $\delta = \min(1 - c_{\min}, \max(\frac{c_{\max}-1}{2}, \frac{1-c_{\min}}{2}))$.

With reroute, each element in the sum of (2.2) is proportional to $\sqrt{\beta(a|s)}|1 - c(s,a)|$ where $c(s,a) \in [c_{\min}, c_{\max}]$. Unlike reroute, other constraints such as the Total Variation (TV), forward and backward KL and PPO were not design to bound the improvement penalty.

2.5 Maximizing the Improvement Step under the Reroute Constraint

We now turn to the problem of maximizing the objective function $J(\pi)$ under the reroute constraint and whether such maximization yields a positive improvement step. Maximizing the objective function without generating new trajectories of π is a hard task since the distribution of states induced by the policy π is unknown. Therefore, usually we maximize a surrogate off-policy objective function $J^{OP}(\pi) = \mathbb{E}_{s\sim\beta}[\sum_a \pi(a|s)A^\beta(s,a)]$. It is common to solve the constrained maximization with a NN policy representation and a policy gradient approach [15, 11]. Here we suggest an alternative: instead of optimizing a parametrized policy that maximizes J^{OP}, the actor (i.e. the agent that interact with the MDP environment) may ad hoc calculate a non-parametrized policy that maximizes the improvement step $\sum_a \pi(a|s)A^\beta(s,a)$ (i.e. the argument of the J^{OP} objective) for each different state. This method maximizes also the J^{OP} objective since the improvement step is independent between states. Note that with an ad hoc maximization, the executed policy is guaranteed to maximize the objective function under the constraint whereas with policy gradient methods one must hope that the optimized policy avoided NN caveats such as overfitting or local minima and converged to the optimal policy.

For the reroute constraint, solving the non-parametrized problem amounts to solving the following simple linear program for each state

$$\begin{aligned} \text{Maximize: } & (\boldsymbol{A}^\beta)^T\boldsymbol{\pi} \\ \text{Subject to: } & c_{\min}\boldsymbol{\beta} \le \boldsymbol{\pi} \le c_{\max}\boldsymbol{\beta} \\ \text{And: } & \sum \pi_i = 1. \end{aligned} \tag{4}$$

Where $\boldsymbol{\pi}$, $\boldsymbol{\beta}$ and $\boldsymbol{A}^\beta$ are vector representations of $(\pi(a_i|s))_{i=1}^{|\mathcal{A}|}$, $(\beta(a_i|s))_{i=1}^{|\mathcal{A}|}$ and $(A^\beta(s,a))_{i=1}^{|\mathcal{A}|}$ respectively. We term the algorithm that solves this maximization problem as Max-Reroute. Similarly, one may derive other algorithms that maximize other constraints.

Notice that Max-Reroute satisfies the conditions of Corollary 2.1.1, therefore it always provides a positive improvement step and hence, at least for a perfect approximation of Q^β it is guaranteed to improve the performance. In addition, notice that Max-Reroute uses only the action ranking information in order to calculate the optimized policy. We postulate that this trait makes it more resilient to value estimation errors. This is in contrast to policy gradient methods which optimize the policy according to the magnitude of the advantage function.

3 Two-armed bandit with Gaussian distributed rewards

To gain some insight into the nature of the RBI step, we examine it in a simplified model of a two-armed bandit with Gaussian distributed rewards [9]. To that end, define the reward of taking action a_i as $r_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and denote action a_2 as the optimal action s.t. $\mu_2 \ge \mu_1$. Consider the learning curve of off-policy learning where a behavior policy is mixed with a fix exploration parameter, i.e. $\beta(a) = \pi(a)(1-\varepsilon) + \frac{\varepsilon}{n_a}$ (where n_a is the number of actions and $\varepsilon = 0.1$). The Q-function is learned with $Q^\pi(a) = (1-\alpha)Q^\pi(a) + \alpha r$, where α is a learning rate, possibly decaying over time. We evaluate several constrained policies: (1) RBI with $(c_{\min}, c_{\max}) = (0.5, 1.5)$, (2) PPO with $\varepsilon = 0.5$, (3) TV with $\delta = 0.25$, (4) greedy step and; (5) forward KL with $\lambda = 1$. RBI, TV and PPO were all maximized with our maximization algorithms (without gradient ascent). To avoid absolute zero probability actions, we clipped the policy such that $\pi(a_i) \ge 10^{-3}$. In addition we added 10 random sample at the start of the learning process. The learning curves are plotted in Figure 1.

The learning curves exhibit two different patterns. For the scenario of $\sigma_1 > \sigma_2$, a fast convergence of all policies was obtained. Essentially, when the better action has low variance it is easy to discard the worse action by choosing it and rapidly improving its value estimation and then switching to the better action. On the other hand, for the case of $\sigma_1 < \sigma_2$ it is much harder for the policy to improve the estimation of the better action after committing to the worse action. We see that RBI defers early commitment and while it slightly reduces the rate of convergence in the first (and easy) scenario, it significantly increases the data efficiency in the harder scenario.

In the second scenario, RBI has the best and KL has the second-best learning curves in terms of initial performance. However, there is another distinction between the ideal learning rate (LR) of $\alpha = \frac{1}{n}$ and a constant rate of $\alpha = 0.01$. In the ideal LR case, the advantage of RBI and KL reduces over time. This is obvious since a LR of $\alpha = \frac{1}{n}$ takes into account the entire history and as such, for large history, after a large number of iterations, there is no need for a policy which learns well from a finite dataset. On the other hand, there is a stable advantage of RBI and KL for a fix LR as fix LR does not correctly weight the entire past experience. Notice that in a larger than 1-step MDP, it is unusual to use a LR of $\frac{1}{n}$ since the policy changes as the learning progress, therefore, usually the LR is fixed or decays over time (but not over state visitations). Hence, RBI has a positive advantage over greedy policies through the entire training process.

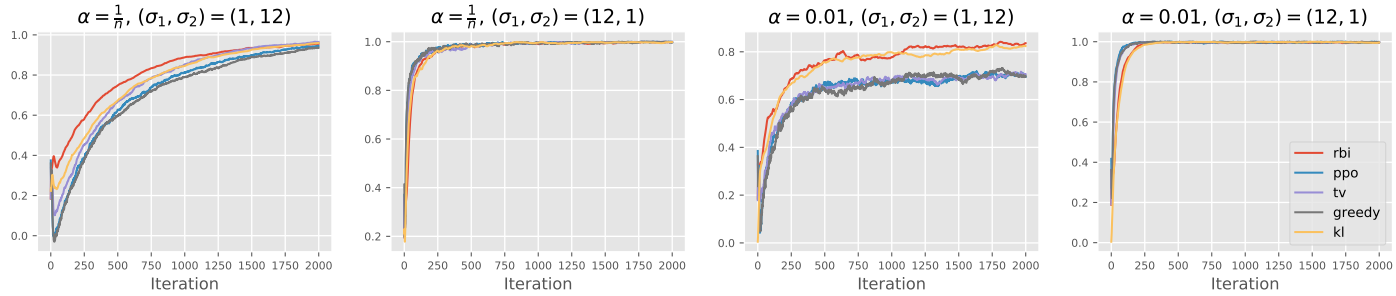


Figure 1: Different constrained policies performances in a two-armed bandit with Gaussian distributed reward setting

4 Experiments in the Atari environment

We implemented an RBI learning agent in the Atari Learning Environment. We adopted a distributed learning setting, similar to the setting of Ape-X [6]. In this experiment we set out to verify: (1) whether RBI is a good approach for Deep RL in terms of better final performance and (2) whether our approach of solving the optimal policy in the non-parametrized space as part of the actor's routine, can be generalized to iterative Deep RL.

To that end, we designed an actor that fetches a stored parametrized policy π_{θ_k} and Q-function $Q^{\pi}_{\phi_k}$. The actor solves the non-parametrized reroute constrained optimization problem (Eq. (4)) and generates an optimized constrained policy π. Our centralized learner imitates the actor's policy with a parametrized policy $\pi_{\theta_{k+1}}$ by minimizing a KL divergence loss $D_{KL}(\pi, \pi_{\theta_{k+1}})$. The learner keeps track of the history Q-function by minimizing the Huber loss $\mathcal{L}(Q^{\pi}_{\phi_{k+1}} - R)$, where R is an n-steps target value $R(s,a) = \sum_{k=0}^{n-1} \gamma^k r_k + \gamma^n \sum_{a'} \pi(s',a') Q^{\pi}_{\bar{\phi}}(s',a')$. Performance curves of 4 Atari games are presented in figure 3 and compared to our Ape-X algorithm implementation. The initial results demonstrate the benefit of using RBI as an efficient general RL learning algorithm.

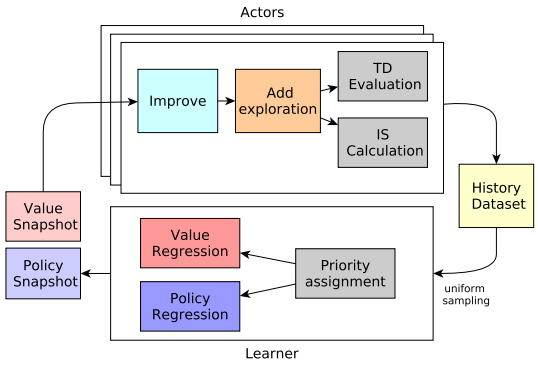


Figure 2: RBI in a distributed RL setting

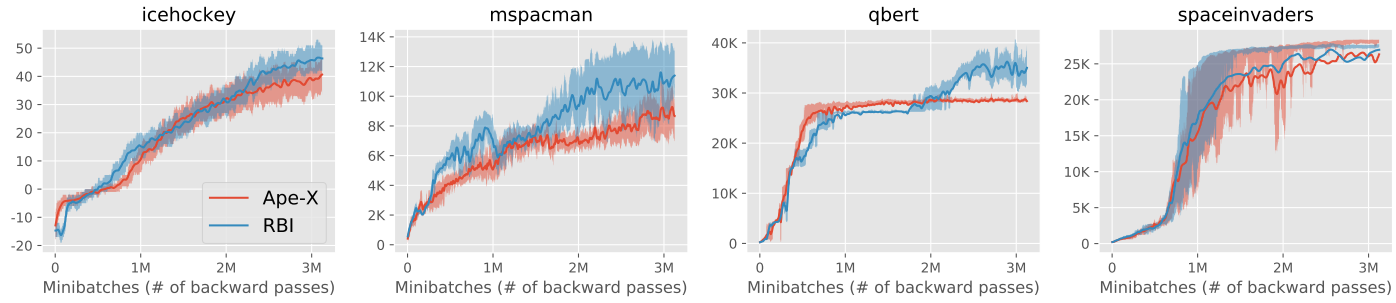


Figure 3: Performance curves of 4 Atari games. The second and third quartiles are shadowed.

References

[1] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[4] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[5] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

[6] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

[7] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

[8] Michael J Kearns and Satinder P Singh. Bias-variance error bounds for temporal difference updates. In *COLT*, pages 142–147. Citeseer, 2000.

[9] Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

[10] OpenAI. Openai five, Jul 2018.

[11] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[14] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[15] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[16] RS Sutton and AG Barto. Reinforcement learning: An introduction, (complete draft), 2017.

[17] Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *International Conference on Machine Learning*, pages 2380–2388, 2015.

[18] Quan Vuong, Yiming Zhang, and Keith W Ross. Supervised policy update for deep reinforcement learning. 2018.

[19] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Does phasic dopamine signalling play a causal role in reinforcement learning?

Peter Shizgal*
Centre for Studies in Behavioural Neurobiology
Department of Psychology
Concordia University
Montréal, QC H4B 1R6
Canada
peter.shizgal@concordia.ca

Ivan Trujillo-Pisanty
Center for the Neurobiology of Addiction, Pain, and Emotion
Department of Anesthesiology and Pain Medicine
Department of Pharmacology
University of Washington
Seattle, WA 98195
ivantp@uw.edu

Marie-Pierre Cossette
Centre for Studies in Behavioural Neurobiology
Department of Psychology
Concordia University
Montréal, QC H4B 1R6
Canada
mpy_cossette@hotmail.com

Kent Conover
Centre for Studies in Behavioural Neurobiology
Department of Psychology
Concordia University
Montréal, QC H4B 1R6
Canada
kentlconover@gmail.com

Francis Carter
Centre for Studies in Behavioural Neurobiology
Department of Psychology
Concordia University
Montréal, QC H4B 1R6
Canada
francis44carter@gmail.com

Vasilis Pallikaras
Centre for Studies in Behavioural Neurobiology
Department of Psychology
Concordia University
Montréal, QC H4B 1R6
Canada
vpallikaras@gmail.com

Yannick-André Breton
Caprion Biosciences
Montréal, QC H2X 3Y7
yannick.breton@gmail.com

Rebecca Brana Solomon
Centre for Studies in Behavioural Neurobiology
Department of Psychology
Concordia University
Montréal, QC H4B 1R6
Canada
rb_solomon@hotmail.com

Abstract

The reward-prediction-error hypothesis holds that payoff from future actions can be maximized and reward predictions optimized by incremental adjustment of connection weights in neural networks underlying expectation and choice. These adjustments are driven by reward prediction errors, discrepancies between the experienced and expected reward. Phasic firing in midbrain dopamine neurons is posited to both represent reward-prediction errors and to cause the weight changes these errors induce. There is abundant correlational evidence from rodents, monkeys, and humans that midbrain dopamine neurons encode reward-prediction errors. The work discussed here tests and challenges the causal component of the reward-prediction-error hypothesis of dopamine activity. Rats were trained to self-administer rewarding electrical stimulation of the medial forebrain bundle or optical stimulation of midbrain dopamine neurons. Stimulation-induced release of dopamine was monitored by means of fast-scan cyclic voltammetry. Both forms of stimulation triggered reliable, recurrent release of dopamine in the nucleus accumbens. According to the RPE-DA hypothesis, such repeated,

*David Munro built and maintained the computer-controlled equipment for experimental control and data acquisition. Software for experimental control and data acquisition was written and maintained by Steve Cabilio. PS web site: http://www.concordia.ca/research/neuroscience/faculty.html?fpid=peter-shizgal. Video re ICSS and the measurement of reward intensity: https://spectrum.library.concordia.ca/978205/

response-contingent release should eventually drive action weights into saturation. If unopposed by a countervailing influence, the repeated release of dopamine should render stable reward-seeking performance at non-maximal levels impossible. Instead, the rats performed at stable non-maximal levels in response to intermediate stimulation strengths.

Keywords: brain stimulation reward; intracranial self-stimulation; medial forebrain bundle; ventral tegmental area; nucleus accumbens; electrical brain stimulation; optogenetics; fast-scan cyclic voltammetry

Acknowledgements

The authors are grateful to Peter Dayan, Ritwik Niyogi, and Sanjeevan Ahilan for many fruitful discussions of the issues addressed here, for their original modeling work on other aspects of the phenomena, and for sharing their insights and interpretations. Steve Cabilio developed and maintained the experimental-control and data acquisition software used in this study. The experimental-control and data acquisition hardware was designed, built, and maintained by David Munro. This work was supported by a Natural Sciences and Engineering Research Council of Canada grant (RGPIN-2016-06703) to PS. The TH-Cre rats were sourced from a colony estabilshed with founders kindly donated by Ilana Witten and Karl Deisseroth.

Reward-prediction errors, dopamine, and intracranial self-stimulation

An elegant, enormously influential hypothesis about the nature and neural mechanisms of learning holds that reward-prediction errors (RPEs), encoded in the firing of dopamine (DA) neurons, optimize expectations about future rewards and the values assigned to reward-seeking actions [1]. The seminal paper introducing this reward-prediction-error hypothesis of dopamine neuron activity (RPE-DA hypothesis) [1] applies temporal-difference reinforcement-learning (TDRL) methods [2] within an actor-critic framework [3].

Paramount in the paper by Montague et al. [1] is their incisive account of DA activity in monkeys performing tasks that combine Pavlovian and operant conditioning. Also included is a brief discussion of how the TDRL model applies to electrical intracranial self-stimulation (eICSS), the performance of instrumental tasks to trigger activation of brain circuitry. The authors point out that DA cell bodies in the ventral tegmental area give rise to axons that course through eICSS sites along the medial forebrain bundle (MFB). Fig. 1 summarizes their portrayal of eICSS, which holds that stimulation of eICSS sites produces a fictive RPE by activating DA neurons.

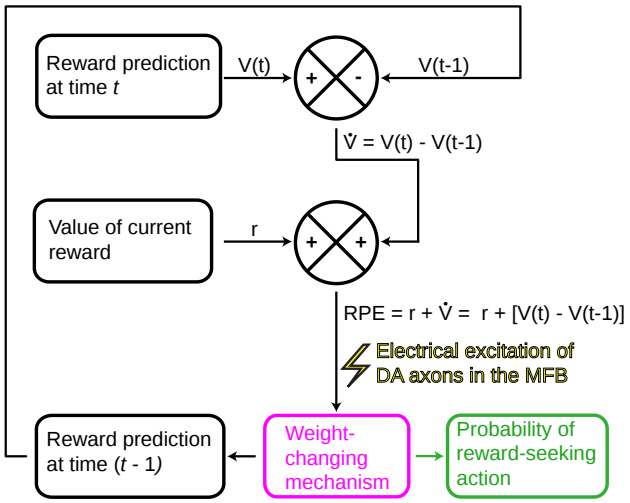


Figure 1: Portrayal of eICSS in [1]

Optogenetic methods make it possible to activate midbrain DA neurons exclusively, unlike electrical stimulation, which is less selective. Rodents will work for such optical stimulation (oICSS) [4, 5, 6]. Specific optical activation of midbrain DA neurons has also been shown to augment responding to a redundant reward-predicting cue that would otherwise have been behaviorally ineffective and to delay extinction of responding to a cue no longer paired with the delivery of a sucrose reward [7]. These results were interpreted as evidence for a causal role of DA-mediated RPEs in learning.

Here, we summarize eICSS and oICSS experiments that reassess the causal role of DA-mediated RPEs in learning. The behavioral and electrochemical findings are not easily explained by the RPE-DA hypothesis.

Two problems with the original TDRL portrayal and a potential remedy

We note two problems with the portrayal in Fig. 1 as it applies to eICSS of the most extensively studied stimulation site: the lateral hypothalamic (LH) level of the MFB. First, the positioning of the electrode isolates the RPE from its corrective consequences. In the case of natural rewards earned under stable conditions, the RPE "predicts itself away." The RPE renders the prediction progressively more accurate, and hence the RPE shrinks progressively to zero. In contrast, an RPE elicited by means of direct axonal stimulation of DA axons in the MFB would be of constant magnitude because it arises beyond the regions where the signals encoding the predicted ($V_t - V_{t-1}$) and experienced (r) rewards must be combined: the somatodendritic region of the DA neurons and/or their afferent network. (Recall that once the value function (V) has been at least partially learned, DA firing is perturbed in opposite directions by these two signals.) If the DA neurons were activated downstream from the point(s) at which these two input signals converge, both the reward prediction and the weight assigned to the reward-seeking action (e.g., lever pressing) would be driven over repeated iterations to their maximal (saturated) values. Stable performance for electrically induced rewards of intermediate magnitude would thus be impossible. This prediction is contradicted by abundant evidence from operant matching experiments on eICSS in which the value of intermediate-strength rewards is stable over many repeated reward encounters, e.g. [8].

The consequences of weight saturation due to stimulation of DA axons are illustrated in panel A of Fig. 2. The agent earns rewards by performing a sustained action ("work"), such as depressing a lever for a required duration. Multiple rewards can be earned during a trial. When the reward is weak ($r = 1$), the latency to begin working is long. However, due to the unconditional RPE, the action weight is boosted by delivery of each reward. Thus, the cumulative work-time accelerates until it attains its maximal velocity. The stronger the reward ($r_5 > r_4 > r_3 > r_2 > r_1$), the shorter the latency to reinitiate responding after reward delivery and the faster the growth of the action weight. Panel B shows the contrasting case of a natural reward. The RPE shrinks as the prediction improves, and the action weights stabilize at values proportional to the reward strength. Thus, the slope of the cumulative work-time trajectory is scaled by the reward strength.

A second problem with the schema in Fig. 1 is the implicit attribution of the behavior to direct activation of DA axons. These small-diameter axons are unmyelinated and have very high thresholds to activation by extracellular currents [9]. Moreover, psychophysical estimates of conduction velocity, recovery from refractoriness, and frequency following in the directly activated MFB fibers subserving eICSS of the MFB implicate neurons with myelinated axons much more readily excited than those of the DA neurons [10].

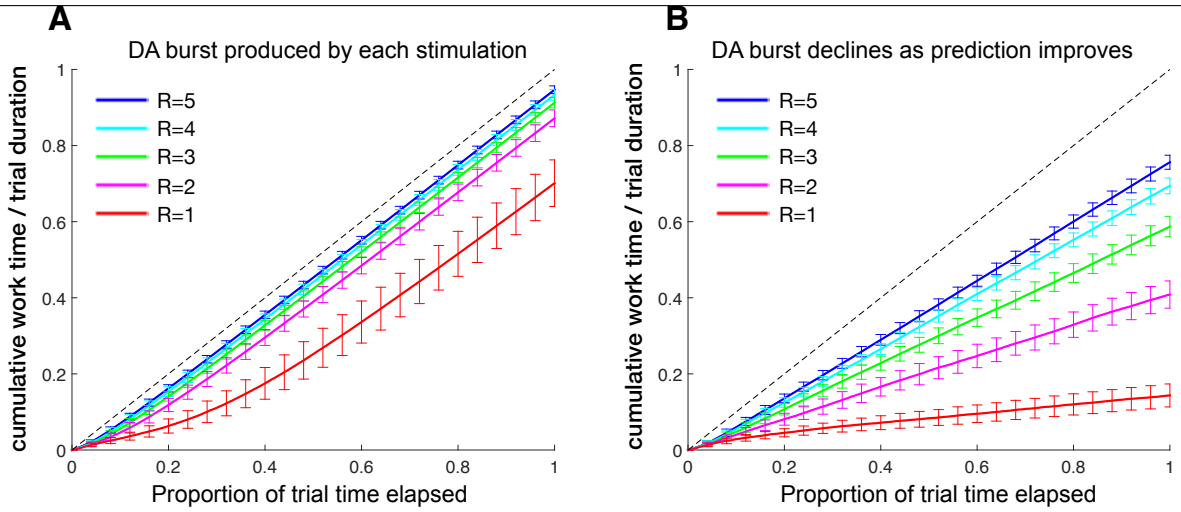


Figure 2: Simulations by Peter Dayan

Both problems could be solved by changing the assumption about where the stimulation intervenes in the TDRL schema (Fig. 3). The rewarding effects of MFB stimulation and intraoral sucrose compete and summate, suggesting that electrical stimulation of the MFB mimics the value of a natural reward [11]. If the electrode indeed excites neurons subserving the primary reward signal (r) instead of those subserving the RPE, then the temporal-difference signal ($V_t - V_{t-1}$) could come to null this electrically induced input, and trajectories such as those shown in panel B of Fig. 2 could be achieved. However, as we show below, this remedy is unavailable in the case of oICSS, which entails direct, unconditional activation of DA neurons. Thus, Fig. 3 predicts: a) oICSS trajectories like those in panel A of Fig. 2 coupled to continued simulation-induced DA release, but b) eICSS trajectories like those in panel B of Fig. 2 coupled to decline and cessation of simulation-induced DA release. We tested these predictions.

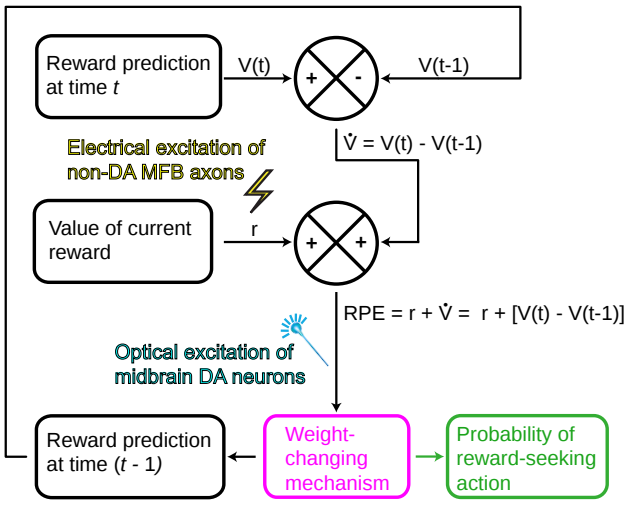


Figure 3: Revised TDRL account of eICSS and oICSS

Methods

ICSS. Electrodes for eICSS were aimed at the LH level of the MFB. Stimulation consisted of 0.5 s trains of constant-current pulses, 0.1 ms in duration. To prepare TH-Cre(+/-) rats for oICSS, channelrhodopsin-2 (ChR2) was expressed in midbrain DA neurons via Cre-Lox recombination and viral transfection, and 300 μm-core optical fibers were aimed at the ventral tegmental area (VTA). Optical stimulation consisted of 1 s trains of 462 or 473 nm pulses, 5 ms in duration.

The triadic-trial paradigm. Experimental sessions were comprised of trials arranged in cycling triads. During the leading trial of each triad, the strength (pulse frequency) of the stimulation was set to the maximum the rat could tolerate, whereas during the trailing trial, it was set to a negligibly rewarding value. The stimulation strength on offer during each central ("test") trial of the triads was also constant within a trial, but it varied across triads, and was selected at random from a vector 3-14 elements in length.The maximum and minimum values of the vector were the strengths used in the leading and trailing trials, respectively.

Electrochemistry. The extracellular DA concentration was measured by means of fast-scan cyclic voltammetry (FSCV). Carbon-fiber microsensors were aimed at the nucleus accumbens (NAc), and an Ag/AgCl reference electrode was positioned 10.7 mm caudal to the NAc. Cyclic voltammograms were generated at 10 Hz by applying an 8.5 ms triangular waveform that ramped from -0.4 V to $+1.3$ V and back to -0.4 V at a scan rate of 400 V/s. A modification of the method of Kishida et al. [12] was used to extract DA concentrations: principal-component regression was substituted for elastic-net regression.

Results and discussion

Fig. 4 shows empirical data from one rat, averaged over 19 sessions, from test trials during which the stimulation strength was sampled randomly from a 14-element vector. Cumulative work time rises at a roughly constant rate, which depends systematically on the strength of the rewarding stimulation. In 22 rats performing eICSS, we obtained 29 datasets

using test-trial stimulation-strength vectors 9 or 14 elements in length. Like those in Fig. 4, the slopes of the cumulative work-time trajectories vary systematically as a function of stimulation strength and are linear or mildly concave downwards. In no case do the data resemble the simulated results in panel A of Fig. 2, which show initial acceleration towards a constant terminal slope due to the unconditional RPE. Instead, the results are consistent with panel B of Fig. 2, in which terminal slopes are related systematically to reward strength, and with the revised TDRL schema in Fig. 3. In the revised schema, stable performance at intermediate levels is achieved because the TD signal can null the input from the stimulation electrode, thus eliminating the RPE and the firing of DA neurons that encodes it. To find out whether DA release indeed ceases during stable eICSS performance at intermediate levels of performance, we measured DA release in the NAc during eICSS by means of FSCV.

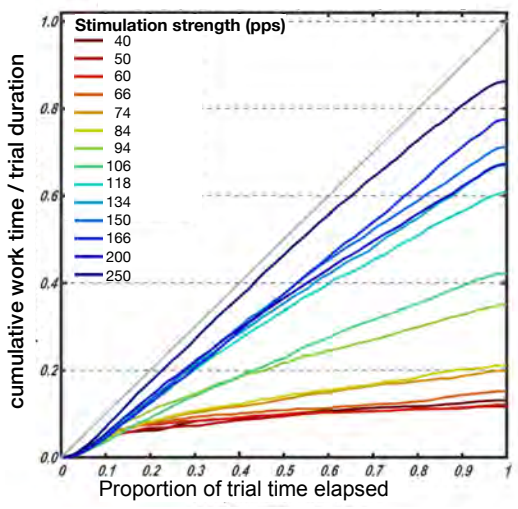


Figure 4: eICSS: stable performance for intermediate-strength electrical rewards

The FSCV recordings were obtained while the rat performed eICSS in a simplified version of the triadic-trial paradigm. Only three stimulation strengths were sampled on test trials: the High and Low values were the same as on leading and trailing trials, respectively, whereas the Med value was intermediate. Behavioral data from one rat, averaged over two test sessions, are shown in panel B of Fig. 5. Again, performance for the medium-strength (Med) reward is roughly stable over the course of the trial. Panel A shows the corresponding measurements of DA concentration, which are cumulations of the peak post-stimulation DA concentration measured following delivery of each stimulation train (panel A of Fig. 6). According to the RPE-DA hypothesis, roughly stable performance at intermediate work levels can be achieved only in the absence of persistent, recurring DA-mediated RPEs. However, panel A of Figs. 5 and 6 show that stimulation-induced DA release continued throughout the eICSS trial, thus calling the hypothesis into question.

Panel B of Fig. 6 shows that optical stimulation of midbrain DA neurons, like electrical stimulation of the MFB (panel A), persistently and reliably elicits transient increases in DA concentration. According to the RPE-DA hypothesis, such transients should alter action weights in the manner depicted in panel A of Fig. 2: when an intermediate-strength reward is on offer during the test trial, the cumulative work trajectory should accelerate until it achieves the maximum slope that the rat's physical capacity allows. This is not what we found.

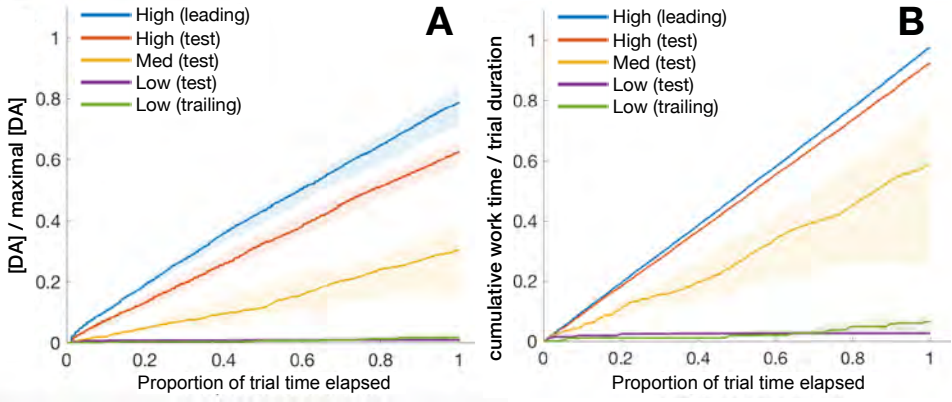


Figure 5: Concurrently acquired FSCV and behavioral data

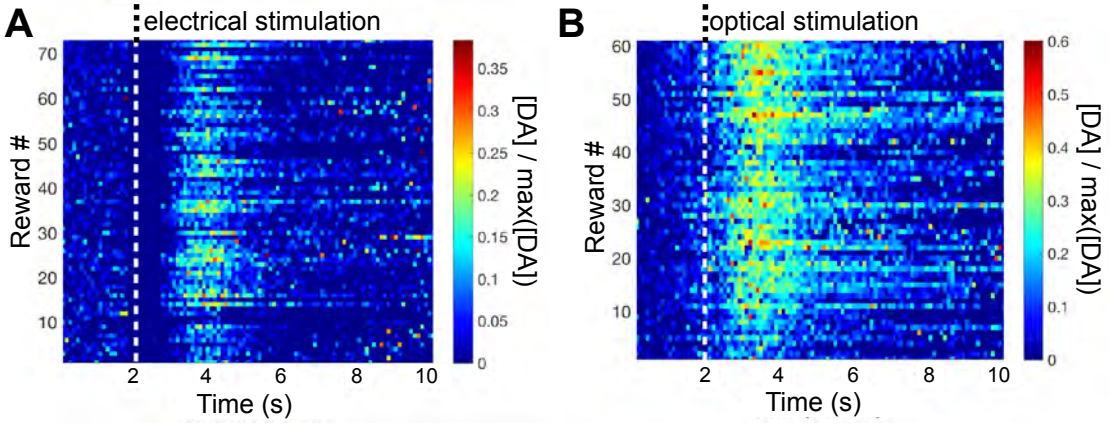


Figure 6: DA transients driven by electrical or optical stimulation

Fig. 7 shows data from a rat working for optical stimulation of midbrain DA neurons in the simplified triadic-trial paradigm. When the reward strength was intermediate, a stable, linear work trajectory is observed. Linear or slightly concave-downward trajectories were also shown by five additional rats performing oICSS in the simplified triadic-trial paradigm. Like the eICSS data, these results call into question a key aspect of the RPE-DA hypothesis, the notion that DA-mediated RPEs cause changes in action weights.

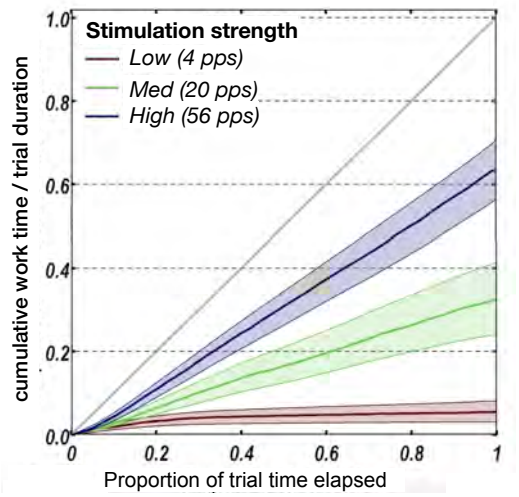


Figure 7: Stable performance for medium-strength optical activation of DA neurons

Reconciling the results with the RPE-DA hypothesis. Peter Dayan has proposed a way to reconcile the present findings with the RPE-DA hypothesis. Could reward predictions come to decrease DA firing in unstimulated neurons, thus compensating for the excitation of the subpopulation of DA neurons recruited by the stimulation? The low baseline firing rate (3-5 spikes s^{-1}) of DA neurons poses a problem for this proposal: the baseline is much closer to zero than to the maximum firing rate. Thus, inhibition of multiple unstimulated DA neurons would be required to compensate for the excitation of each stimulated neuron. This would be difficult to achieve given the massive, bilateral recruitment of midbrain DA neurons by electrical MFB stimulation. That said, this proposal merits rigorous experimental test.

Limitations. In the different versions of the triadic-trial paradigm, stable behavioral data has been obtained from 26 rats performing working for electrical stimulation and 9 rats working for optical stimulation. However, concurrent measurements of behavior and DA concentration have been carried out successively in only two rats to date. Additional subjects must be tested, and the FSCV recording sites must be adjusted in the light of recent findings showing functional specialization of NAc subregions [13].

Conclusion. The findings reported here raise serious questions about the causal component of the RPE-DA hypothesis and provide several proofs of principle for novel ways to test this foundational idea.

References

[1] P. R. Montague, P. Dayan, & T. J. Sejnowski, "A framework for mesencephalic dopamine systems based on predictive Hebbian learning", *The Journal of neuroscience*, vol. 16, no. 5, pp. 1936–1947, 1996.

[2] R. S. Sutton, "Learning to predict by the methods of temporal differences", *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[3] J. C. Houk, J. L. Adams, & A. G. Barto, "A model of how the basal ganglia generate and use neural signals that predict reinforcement", in J. C. Houk, J. L. Davis, & D. G. Beiser (Eds.), *Models of information processing in the Basal Ganglia*, pp. 249–270, The MIT Press, 1995.

[4] A. R. Adamantidis, H.-C. Tsai, B. Boutrel, *et al.*, "Optogenetic Interrogation of Dopaminergic Modulation of the Multiple Phases of Reward-Seeking Behavior.", *The Journal of neuroscience*, vol. 31, no. 30, pp. 10829–10835, 2011.

[5] C. D. Fiorillo, "Transient activation of midbrain dopamine neurons by reward risk", *Neuroscience*, vol. 197, no. C, pp. 162–171, 2011.

[6] I. B. Witten, E. E. Steinberg, T. J. Davidson, *et al.*, "Recombinase-driver rat lines: tools, techniques, and optogenetic application to dopamine-mediated reinforcement.", *Neuron*, vol. 72, no. 5, pp. 721–733, 2011.

[7] E. E. Steinberg, R. Keiflin, J. R. Boivin, *et al.*, "A causal link between prediction errors, dopamine neurons and learning", *Nature Neuroscience*, vol. 16, pp. 966–973, 2013.

[8] M. I. Leon, V. Rodriguez-Barrera, & A. Amaya, "The effect of scopolamine on matching behavior and the estimation of relative reward magnitude.", *Behavioral Neuroscience*, vol. 131, no. 5, pp. 406–420, 2017.

[9] J. S. Yeomans, N. T. Maidment, & B. S. Bunney, "Excitability properties of medial forebrain bundle axons of A9 and A10 dopamine cells.", *Brain research*, vol. 450, no. 1-2, pp. 86–93, 1988.

[10] P. Shizgal, "Neural basis of utility estimation", *Current Opinion in Neurobiology*, vol. 7, no. 2, pp. 198–208, 1997.

[11] K. L. Conover & P. Shizgal, "Competition and summation between rewarding effects of sucrose and lateral hypothalamic stimulation in the rat", *Behavioral Neuroscience*, vol. 108, no. 3, pp. 537–548, 1994.

[12] K. T. Kishida, I. Saez, T. Lohrenz, *et al.*, "Subsecond dopamine fluctuations in human striatum encode superposed error signals about actual and counterfactual reward", *Proceedings of the National Academy of Sciences*, vol. 113, no. 1, pp. 200–205, 2016.

[13] J. W. de Jong, S. A. Afjei, I. Pollak Dorocic, *et al.*, "A Neural Circuit Mechanism for Encoding Aversive Stimuli in the Mesolimbic Dopamine System", *Neuron*, vol. 101, no. 1, pp. 133–151.e7, 2019.

Robust Pest Management Using Reinforcement Learning

Talha Siddique
Department of Natural Resources and the Environment
University of New Hampshire
Durham, NH 03824
ts1121@wildcats.unh.edu

Jia Lin Hau
Department of Computer Science
University of New Hampshire
Durham, NH 03824
jh1111@wildcats.unh.edu

Shadi Atallah
Department of Natural Resources and the Environment
University of New Hampshire
Durham, NH 03824
shadi.atallah@unh.edu

Marek Petrik
Department of Computer Science
University of New Hampshire
Durham, NH 03824
mpetrik@cs.unh.edu

Abstract

Developing effective decision support systems for agriculture matters. Human population is likely to peak at close 11 billion and changing climate is already reducing yields in the Great Plains and in other fertile regions across the world. With virtually all arable land already cultivated, the only way to feed the growing human population is to increase yields. Making better decisions, driven by data, can increase the yield and quality of agricultural products and reduce their environmental impact.

In this work, we address the problem of an apple orchardist who must decide how to control the population of codling moth, which is an important apple pest. The orchadist must decide when to apply pesticides to optimally trade off apple yields and quality with the financial and environmental costs of using pesticides. Pesticide spraying decisions are made weekly throughout the growing season, with the yield only observed at the end of the growing season. The inherent stochasticity driven by weather and delayed rewards make this a classical reinforcement learning problem.

Deploying decision support systems in agriculture is challenging. Farmers are averse to risk and do not trust purely data-driven recommendations. Because weather varies from season to season and ecological systems are complex even a decade worth of data may be insufficient to get good decisions with high confidence.

We propose a robust reinforcement learning approach that can compute good solutions even when the models or rewards are not known precisely. We use Bayesian models to capture prior knowledge. Our main contribution is that we evaluate which model and reward uncertainties have the greatest impact on solution quality.

Keywords: Natural Resources, Small Data, Markov Decision Process, Robust Optimization.

1 Introduction

Apple is the most consumed fruit in the USA. In 2015, an average person in the USA consumed about 115.4 pounds of fresh and processed fruits and 24.7 pounds of apple in other forms such as juice, canned, frozen etc (USDA ERS, 2015). One of the most problematic apple pests is the codling moth (Cydia Pomonella). If left unchecked, it can claim up to 95 percent of the seasons apple crops [8]. Common pest management actions are mostly dependent on chemical pesticides. Avoiding the use of a pesticide improves the quality of an apple crop and increases profits but waiting too long after the pest is detected can lead to a crop failure [4, 10, 11].

There is a need for effective decision support tools. However, developing cheap and effective strategies can be challenging because natural systems are complex, difficult to model, and expensive to observe [6]. Recently, reinforcement learning has been used with great success to model such complex domains [7]. The drawback of using such techniques is that they require data sets of appropriate size and accuracy. Even though thousands of samples can be generated using domain simulators but any form of available data sets on the distribution of invasive species like pests tend to suffer from biases and inaccuracy [5]. To make more effective and timely decisions, there is a need for decision support systems that recommend safe actions in the face of limited and flawed data [1].

In this paper, we address the problem of an orchard manager who must decide, given a pest population level, whether to apply pesticides, in order to maximize the value of the orchard over a finite time horizon. We can formulate the management problem using the reinforcement learning methodology. The pesticide application policies must be based on data consisting of imperfect observations of the pest population level. For our solutions to be immune to such data and parameter uncertainties, there is a need for our method to be robust. A robust method will compute solutions that trade-off brittle optimality for increased confidence.

We develop and evaluate a new method for robust reinforcement learning, which can reliably compute pest management policies from imperfect observational data. They can determine pest control policies that are likely to work well even if observations of pest population levels are scattered and does not resemble the true distribution accurately. To address such uncertainties from data and parameters in a tractable way, we combine the flexibility of Bayesian modeling with the computational tractability of robust optimization. We will be using the Bayesian modeling tool Stan to implement our approach.

Unfortunately, planning directly with posterior distributions leads to intractable optimization problems [2, 3, 9]. We are, instead, proposing to use the posterior distributions to construct the ambiguity set and then use tractable robust optimization methods.

The contributions of this paper is to advance the understanding of uncertainty in reinforcement learning. Our research provides new insights into the benefits and drawbacks of considering uncertainty given a specific problem structure.

2 Experiments and Results

Given our model, we compare the performance of different pesticide application schedules using threshold policies. A threshold policy applies the pesticide only when the pest level exceeds the given threshold.

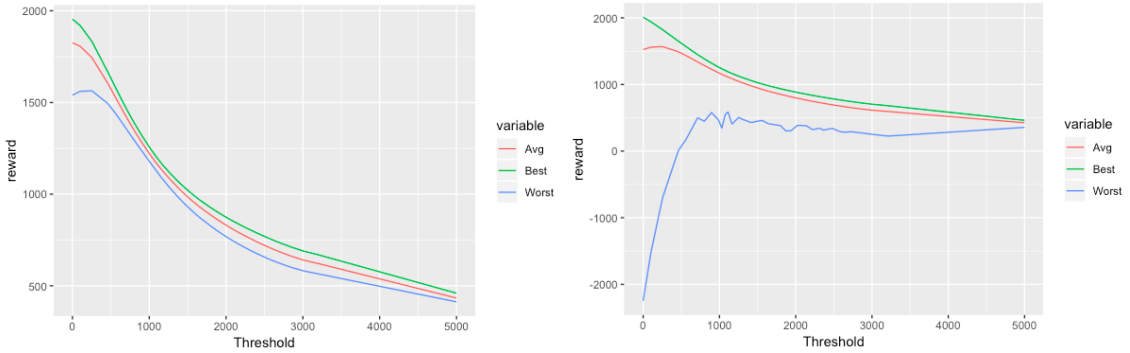
For each threshold policy π, we determine the best case, worst case, and average case scenario returns, when we apply it to the set of different pest growth rates: λ. The pest growth rates λ are sampled from our Bayesian model. In addition, we will also determine the robust return given our π and λ.

Let $\rho(\pi, \lambda)$ be the return of policy π under the growth rate λ. Following are the mathematical definitions of each of our objectives:

1. Best case scenario return: $\max_{\lambda,\epsilon,r} \rho(\pi, \lambda)$
2. Worst case scenario return: $\min_{\lambda,\epsilon,r} \rho(\pi, \lambda)$
3. Average case scenario return: $\mathbb{E}_{\lambda,\epsilon,r}\rho(\pi, \lambda)$
4. Robust return: $\max_{\pi} \min_{\lambda,\epsilon,r} \rho(\pi, \lambda)$

According to Figure 1(a), we plot the graph where we only consider the uncertainty of the growth rate λ. Since the best, worst, and average cases have similar optimal threshold policies, the robustness does not really make a difference compared to the regular approach. All objectives suggest the farmer to always spray the plant because the lower threshold has a higher reward. The intuition of the policy is, if a farmer always sprays pesticide to their apples, the number of moths will be the least and they will likely to get a good apple and a high reward (revenue).

However, this is not realistic, spraying pesticides frequently could damage the soil or your plant which is the cost that our model does not capture. The second issue of the simulation and our model did not capture, is when an apple turns bad, some of the damage of an apple is not reversible, we could kill the moth in the apple but the apple will still remain damage. Therefore, to make the reward function more realistic, we included 2 variables (i) cost of spraying and



(a) Uncertainty of Growth Rate λ (b) Added Uncertainty of Cost and Penalty

Figure 1: Returns as a Function of Application Threshold

(ii) irreversible financial penalty for infected apples. Since we do not know the real underlying cost for spraying and irreversible penalty we created a distribution of possible cost and explore the effect of the cost.

In Figure 1(b), we depict the costs of spraying and an irreversible penalty for a bad apple to make the model more realistic. After we added the cost of spraying and penalty for irreversible apple damage, we can see that the robust policy which maximizes the worst case and the regular approach which maximize the average case are totally different to each other.

3 Conclusion

To conclude, robust optimization is not always necessary if we have uncertainty on a linear function. However, robust optimization is crucial if we have uncertainty in other types of function. From our experiment, we show the importance to identify and define the uncertainties of all important factors in a model, which otherwise could lead to a very different solution. An example is the solutions determined by our model where after defining the cost of spraying and penalty for infected apples, we can see an obvious change in policy.

References

[1] L R Carrasco, R Baker, a Macleod, J D Knight, and J D Mumford. Optimal and robust control of invasive alien species spreading in homogeneous landscapes. *Journal of the Royal Society, Interface / the Royal Society*, 7(September):529–540, 2010.

[2] Michael Castronovo, Damien Ernst, Adrien Couëtoux, and Raphael Fonteneau. Benchmarking for Bayesian reinforcement learning. *PLoS ONE*, 11(6):1–25, 2016.

[3] Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[4] Darwin C. Hall and Richard B. Norgaard. On the Timing and Application of Pesticides: Reply. *American Journal of Agricultural Economics*, 56(3):644, 2006.

[5] Koji Kotani, Makoto Kakinaka, and Hiroyuki Matsuda. Optimal invasive species management under multiple uncertainties. *Mathematical Biosciences*, 233(1):32–46, 2011.

[6] Andreas Lydakis, Jenica M. Allen, Marek Petrik, and Tim M. Szewczyk. Robust strategies for managing invasive plants. In *IJCAI Workshop: Artificial Intelligence for Wildlife Conservation*, 2018.

[7] Matthew H. Meisner, Jay A. Rosenheim, and Ilias Tagkopoulos. A data-driven, machine learning framework for optimal pest management in cotton. *Ecosphere*, 7(March):1–13, 2016.

[8] M.Gregory Peck and Ian A. Merwin. A Grower ' s Guide to Organic Apples. *Agriculture*, 208(NYS IPM Publication No 223):43–56, 2016.

[9] Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An Analytic Solution to Discrete Bayesian Reinforcement Learning. In *International Conference of Machine Learning (ICML)*, 2006.

[10] W. D. Shaw. Environmental and Natural Resource Economics: Decisions Under Risk and Uncertainty. *Encyclopedia of Energy, Natural Resource, and Environmental Economics*, 3-3(979):10–16, 2013.

[11] Richard T. Woodward and David Tomberlin. Practical Precautionary Resource Management Using Robust Optimization. *Environmental Management*, 54(4):828–839, 2014.

Learning Powerful Policies by Using Consistent Dynamics Model

Shagun Sodhani *
Mila
University of Montreal

Anirudh Goyal
Mila
University of Montreal

Tristan Deleu
Mila
University of Montreal

Yoshua Bengio
CIFAR Senior Fellow and Mila
University of Montreal

Sergey Levine
University of California
Berkeley

Jian Tang
Mila
University of Montreal

Abstract

Model-based Reinforcement Learning approaches have the promise of being sample efficient. Much of the progress in learning dynamics models in RL has been made by learning models via supervised learning. There is enough evidence that humans build a model of the environment, not only by observing the environment but also by interacting with the environment. Interaction with the environment allows humans to carry out *experiments*: taking actions that help uncover true causal relationships which can be used for building better dynamics models. Analogously, we would expect such interactions to be helpful for a learning agent while learning to model the environment dynamics. In this paper, we build upon this intuition, by using an auxiliary cost function to ensure consistency between what the agent observes (by acting in the real world) and what it imagines (by acting in the "learned" world). We consider several tasks - Mujoco based control tasks and Atari games - and show that the proposed approach helps to train powerful policies and better dynamics models.

Keywords: Reinforcement Learning, Deep Reinforcement Learning, Model-Free RL, Model-Based RL,

Acknowledgements

The authors acknowledge the important role played by their colleagues at Mila throughout the duration of this work. The authors would like to thank Bhairav Mehta and Gautham Swaminathan for their feedback on the initial manuscript. The authors are grateful to NSERC, CIFAR, Google, Samsung, Nuance, IBM, Canada Research Chairs, Canada Graduate Scholarship Program, Nvidia for funding, and Compute Canada for computing resources. We are very grateful to Google for giving Google Cloud credits used in this project.

*Corresponding author: sshagunsodhani@gmail.com

1 Introduction

Reinforcement Learning (RL) consists of two fundamental problems: *learning* and *planning*. *Learning* refers to improving the agent's policy by interacting with the environment while *planning* refers to improving the policy without interacting with the environment. These problems evolve into the dichotomy of *model-free* methods (which primarily rely on *learning*) and *model-based* methods (which primarily rely on *planning*). While *model-free* methods have shown many successes [1, 2, 3], their high sample complexity remains a major challenge. In contrast, model-based RL methods aim to improve the sample efficiency by learning a dynamics model of the environment. But these methods have several caveats. If the policy takes the learner to an unexplored state in the environment, the learner's model could make errors in estimating the environment dynamics, leading to sub-optimal behavior. This problem is referred to as the model-bias problem [4].

To make predictions about the future, dynamics models are unrolled step by step leading to "compounding errors" [5, 6]: an error in modeling the environment at time t affects the predicted observations at all subsequent steps. In the model-based approaches, the dynamics model is usually trained with supervised learning techniques and the state transition tuples (collected as the agent acts in the environment) become the supervising dataset. Hence the process of learning the model has no control over what kind of data is produced for its training. That is, from the perspective of learning the dynamics model, the agent just observes the environment and does not "interact" with it. On the other hand, there's enough evidence that humans learn the environment dynamics not just by observing the environment but also by interacting with the environment [7, 8]. Interaction is useful as it allows the agent to carry out experiments in the real world which is clearly a desirable characteristic when building dynamics models.

This leads to an interesting possibility. Consider a learning agent training to optimize an expected returns signal in a given environment. At a given timestep t, the agent is in some state $s_t \in S$ (State space). It takes an action $a_t \in A$ (action space) according to its policy $a_t \sim \pi_t(a_t|s_t)$, receives a reward r_t (from the environment) and transitions to a new state s_{t+1}. The agent is trying to maximize its expected returns and has two pathways for improving its behaviour:

1. ***Close-loop* path:** The agent interacts with the environment acting in the real world at every step. The agent starts in state s_0 and is in state s_t at time t. It chooses an action a_t to perform (using its policy π_t), performs the chosen action, and receives a reward r_t. It then observes the environment to obtain the new state s_{t+1}, uses this state to decide which action a_{t+1} to perform next and so on.
2. ***Open-loop* path:** The agent interacts with the learned dynamics model by *imagining* to act and predicts the future observations (or future belief state in case of state space models). The agent starts in state s_0 and is in state s_t at time t. Note that the agent "imagines" itself to be in state s^I_t and can not access the true state. It chooses an action a_t to perform (using its policy π_t), acts in the "learner's" world (dynamics model) and imagines to transition to the new state s^I_{t+1}. The current "imagined" state is used to predict the next "imagined" state. During these "imagined" roll-outs, the agent only interacts with the learner's "world" and not with the environment.

The agent could use both the pathways simultaneously. It could, in parallel, (i) build a model of the environment (*dynamics* model) and (ii) engage in interaction with the real environment as shown in Figure 1. As such, the two pathways may not be *consistent* given the challenges in learning a multi-step dynamics model. By *consistent*, we mean the behavior of state transitions along the two paths should be indistinguishable. Had the two pathways would be *consistent* and we could say that the learner's dynamics model is grounded in reality. To that end, our contributions are the following:

1. We propose to ensure consistency by using an auxiliary loss which explicitly matches the generative behavior (from the open loop) and the observed behavior (from the closed loop) as closely as possible.
2. We evaluate our approach on 7 Mujoco based continuous control tasks and 4 Atari games and observe that the proposed approach helps to train more powerful policies.
3. We compare our proposed approach to the state-of-the-art state space models [9] and show that the proposed method outperforms the sophisticated baselines despite being very straightforward.

2 Consistency Constraint

We impose the consistency constraint by encoding the state transitions (during both open-loop and closed-loop) into fixed-size real vectors using recurrent networks and enforce the output of the recurrent networks to be similar in the two cases. Encoding the sequence can be seen as abstracting out the per-step state transitions into how the dynamics of the environment evolve over time. This way, we do not focus on mimicking each state but the high-level dynamics of the state transitions and the dynamics model focuses only on information that makes the multi-step predictions indistinguishable from the actual observations from the environment (figure 1). We minimize the $L2$ error between the encoding of predicted future observations as coming from the learner's dynamics model (during open-loop) and the encoding of the future observations as coming from the environment (during closed loop).

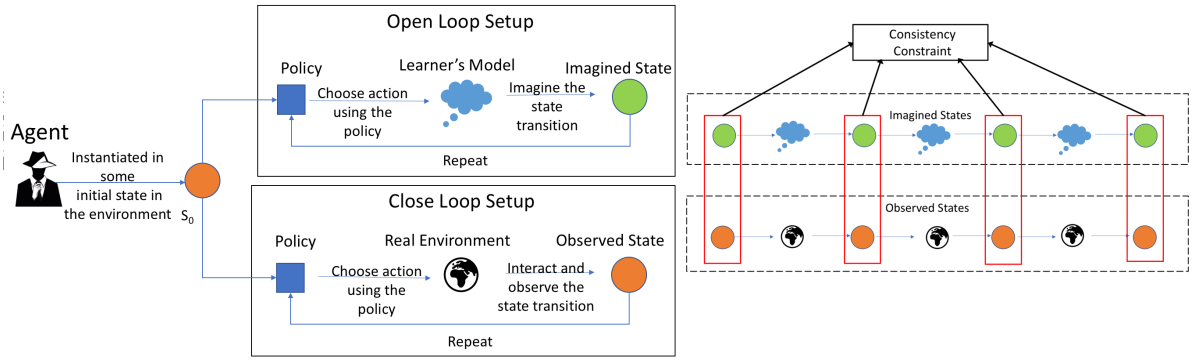


Figure 1: The agent, in parallel, (i) Builds a model of the world and (ii) Engages in an interaction with the world. The agent can now learn the model dynamics while interacting with the environment. We show that making these two pathways consistent helps in simultaneously learning a better policy and a more powerful generative model.

Let us assume that the agent started in state s_0 and that $a_{0:T-1}$ denote the sequence of actions that the agent takes in the environment from time $t = 0$ to $T-1$ resulting in state sequence $s_{1:T}$ that the agent transitions through. Alternatively, the agent could have "imagined" a trajectory of state transitions by performing the actions $a_{0:T-1}$ in the learner's dynamics model. This would result in the sequence of states $s^I_{1:T}$. The consistency loss is computed as follows:

$$l_{cc}(\theta, \phi) = \|enc(s_{1:T})) - enc(s^I_{1:T}))\| \tag{1}$$

where $\|\|$ denotes the L2 norm, $enc(s_{1:T})) = RNN([s_1, s_2, ..., s_T])$ and $enc(s^I_{1:T})) = RNN([s^I_1, s^I_2, ..., s^I_T])$. The agent which is trained with the *consistency constraint* is referred to as the *consistent dynamics* agent. The overall loss for such a learning agent can be written as follows:

$$l_{total}(\theta, \phi) = l_{rl}(\theta, \phi) + \alpha l_{cc}(\theta, \phi) \tag{2}$$

θ refers to the parameters of the agent's transition model $\hat{f}$ and ϕ refers to the parameters of the agent's policy π. The first component, $l_{rl}(\theta, \phi)$, corresponds to the RL objective i.e maximizing expected return and is referred to as the *RL loss*. The second component, $l_{cc}(\theta, \phi)$, corresponds to the loss associated with the *consistency constraint* and is referred to as *consistency loss*. α is a hyper-parameter to scale the *consistency loss component* with respect to the *RL loss*.

We consider both observation space models (where the environment is modeled as a Markov Decision Process) and state-space models where the learning agent encodes the observation into a high-dimensional latent space. State space models are useful when the observation space is high dimensional, as in case of pixel-space observations. For the state space models, the agent learns to model the environment dynamics in the latent space.

3 Rationale Behind Using Consistency Loss

Our goal is to provide a mechanism for the agent to have a direct "interaction" between the policy and the dynamics model. This interaction is different from the standard RL approaches where the trajectories sampled by the policy are used to train the dynamics model. In those cases, the model has no control over what kind of data is produced for its training and there is no ("direct") mechanism for the dynamics model to affect the policy, hence a "direct interaction" between the policy and the model is missing. A practical instantiation of this idea is the *consistency loss* where we ensure consistency between the predictions (from the dynamics model) and the actual observations (from the environment). This simple baseline works surprisingly well compared to the state-of-the-art methods (as demonstrated by our experiments).

Our approach is different from just learning a k-step prediction model as in our case, we have two learning signals for the policy: The one from the reinforcement learning loss (to maximize return) and the other due to consistency constraint. This provides a mechanism where learning a model can itself change the policy (thus "interacting" with the policy). In the standard case, the state transition pairs (collected as the agent acts in the environment) become the supervising dataset for learning the model and there is no feedback from the model learning process to the policy.

4 Experimental Results

We evaluate how well does the proposed *Consistent Dynamics* model compares against the state-of-the-art approaches for observation space models and state space models. All the results are reported after averaging over 3 random seeds. Note that our simplistic approach outperforms the state-of-the-art *Learning to Query* model [9].

4.1 Observation Space Models

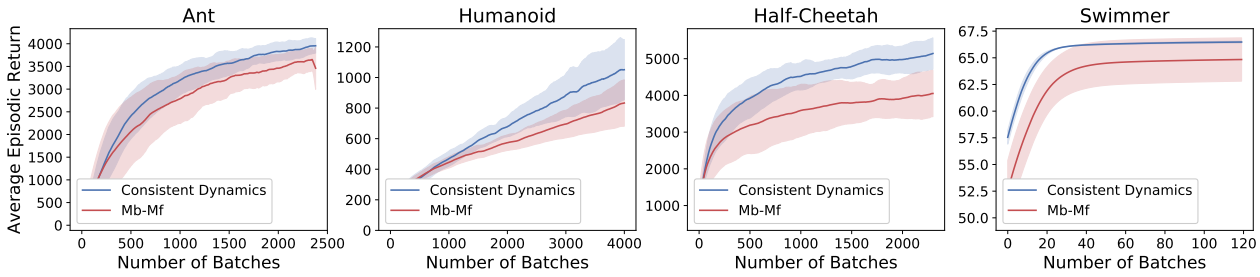


Figure 2: Comparison of the average episodic returns, for *Mb-Mf* agent and *consistent dynamics* agent on the Ant, Humanoid, Half-Cheetah and Swimmer environments (respectively). Note that the results are averaged over 100 batches for Ant, Humanoid and Half-Cheetah and 10 batches for Swimmer.

We use the hybrid Model-based and Model-free (*Mb-Mf*) algorithm [10] as the baseline model. The policy and the dynamics model are learned jointly. We consider 4 Mujoco environments from RLLab [11]: Ant ($S \in R^{41}$, $A \in R^{8}$), Humanoid ($S \in R^{142}$, $A \in R^{21}$), Half-Cheetah ($S \in R^{23}$, $A \in R^{6}$) and Swimmer ($S \in R^{17}$, $A \in R^{3}$). For computing the consistency loss, the learner's dynamics model is unrolled for $k = 20$ steps and GRU model is used[12].

Figure 2 compares the average episodic returns for the baseline *Mb-Mf* model (does not use consistency loss) and the proposed *consistent dynamics* model (*Mb-Mf* model + consistency loss). Using consistency helps to learn a better policy in fewer updates for all the environments. We also study the effect of changing k (during training) ($k \in \{5, 10, 20\}$) and observe that a higher value of k ($k = 20$) leads to better returns for all the tasks.

4.2 State Space Models

We use the state-of-the-art *Learning to Query* model [9] as the baseline. We train an expert policy for sampling high-reward trajectories which are used to train the policy (using imitation learning) and the dynamics model (by max-likelihood). We consider 3 continuous control tasks from the OpenAI Gym suite [13]: Half-Cheetah, Fetch-Push and Reacher. During the open loop, the dynamics model is unrolled for $k = 10$ steps for Half-Cheetah and $k = 5$ for other tasks.

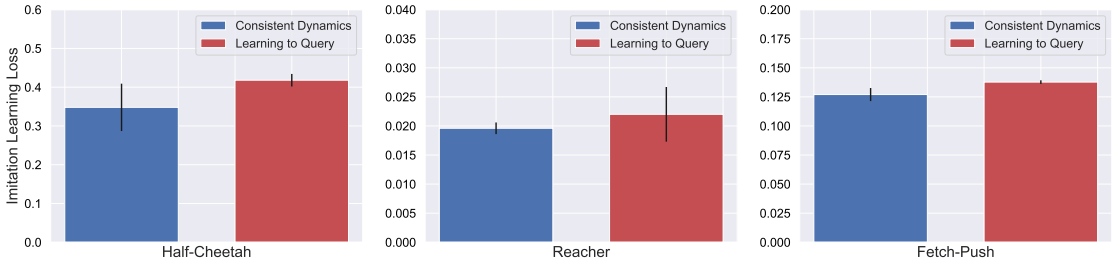
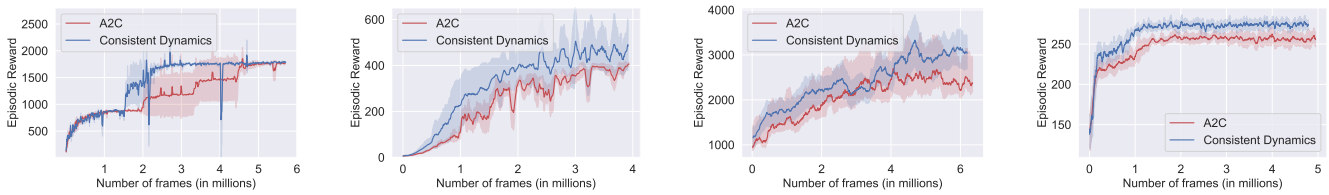


Figure 3: Comparison of imitation learning loss for the *Consistent Dynamics* agent (ie *Learning to Query* agent + consistency loss) and the baseline (just *Learning to Query*) for Half-Cheetah and Reacher environments. The plot for Fetch-Push environment is in the appendix. The bars represents the values corresponding to the trained agent, averaged over the last 50 batches of training. Using consistency constraint leads to a more powerful policy.

In figure 3, we compare the imitation learning loss for the *Consistent Dynamics* agent (*Learning to Query* agent with consistency loss) and the baseline (*Learning to Query* agent) and show that consistency constraint helps to learn a more powerful policy. Sampling from recurrent dynamics model is prone to *compounding errors* as even small prediction errors can compound when sampling for a large number of steps. We evaluate the robustness of the proposed model by unrolling the model for much longer (50 timesteps) than it was trained on (10 timesteps). We observe that the auxiliary cost (which is not solely focused on predicting the next observation) helps to learn a better model

4.3 Atari Environment

We evaluate the proposed approach on Atari games [14] using A2C as the baseline model and by adding consistency loss to A2C to obtain the *Consistent Dynamics* model. Specifically, we consider four environments - Seaquest, Break-



(a) Seaquest (b) Breakout (c) MsPacman (d) Frostbite

Figure 4: Comparison of average episodic return on four Atari environments (Seaquest, Breakout, MsPacman and Frostbite respectively), for the *Consistent Dynamics* agent (ie *A2C* agent + consistency loss) and the baseline (just *A2C*). Using consistency constraint leads to a more powerful policy. Note that the results are average over 100 episodes.

out, MsPacman, and Frostbite and show that the proposed approach is more sample efficient as compared to the A2C approach thus demonstrating the applicability of our approach to different environments and learning algorithms.

5 Conclusion

In this work, we formulate a way to ensure consistency between the predictions of a dynamics model and the real observations from the environment thus allowing the agent to learn powerful policies. We apply an auxiliary loss which encourages the state transitions in the actual environment to be indistinguishable from state transitions in the learner's dynamics model. We consider both observation space and state space models and show that the proposed method outperforms the sophisticated baselines despite being quite simple.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[2] V. Mnih, A. Puigdomènech Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *ArXiv e-prints*, February 2016.

[3] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. *ArXiv e-prints*, February 2015.

[4] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[5] Erik Talvitie. Model regularization for stable sample rollouts. In *UAI*, 2014.

[6] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. *ArXiv e-prints*, October 2016.

[7] Claire Cook, Noah D Goodman, and Laura E Schulz. Where science starts: Spontaneous experiments in preschoolers exploratory play. *Cognition*, 120(3):341–349, 2011.

[8] Bryan C Daniels and Ilya Nemenman. Automated adaptive inference of phenomenological dynamical models. *Nature communications*, 6:8133, 2015.

[9] L. Buesing, T. Weber, S. Racaniere, S. M. A. Eslami, D. Rezende, D. P. Reichert, F. Viola, F. Besse, K. Gregor, D. Hassabis, and D. Wierstra. Learning and Querying Fast Generative Models for Reinforcement Learning. *ArXiv e-prints*, February 2018.

[10] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. *ArXiv e-prints*, August 2017.

[11] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. *ArXiv e-prints*, April 2016.

[12] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *ArXiv e-prints*, June 2014.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[14] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

An Attractor Neural-Network for Binary Decision Making

Ashley Stendel
Department of Computer Science
McGill University
ashley.stendel@mail.mcgill.ca

Thomas R. Shultz
Department of Psychology
McGill University
thomas.shultz@mcgill.ca

Abstract

We apply an attractor neural-network model to experiments on monkeys who decide which direction tokens are moving, while firing rates of large numbers of neurons in premotor cortex are being recorded. Using pools of artificial excitatory and inhibitory neurons, our network model accurately simulates the neural activity and decision behavior of the monkeys. Among the simulated phenomena are decision time and accuracy, commitment, patterns of neural activity in trials of varying difficulty, and an urgency signal that builds over time and resets at the moment of decision.

Keywords: Decision making; computer simulation; attractor neural-networks; inhibition and excitation; urgency.

Acknowledgements

This work was supported in part by grants from NSERC (Natural Sciences and Engineering Research Council of Canada RGPIN/7927-2012) and UNIQUE (Unifying Neuroscience and artificial Intelligence in Quebec 2020-RS4-265502). We thank Dr. Paul Cisek for his feedback and pointers.

1 Introduction

We focus on simulating experiments tracking neural firing in a pair of Macaque monkeys who were trying to decide the predominant direction in which a population of 15 jumping tokens were moving, to the left or right, under three levels of trial difficulty (Thura & Cisek, 2014). The monkeys were rewarded with juice if they correctly predicted the direction. Neural activity recordings were taken from individual neurons in the dorsal premotor (PMd) and primary motor cortex (M1). During deliberation, activity in both cortical regions tracked token movement information and combined it with a growing urgency signal that predicted the end of a trial. Neural activity responded to changes in token jumps. Approximately 280 ms before reach onset, PMd activity tuned to the selected target reached a consistent peak while activity tuned to the unselected target was suppressed.

Easy, ambiguous, and misleading trials were classified post hoc from fully random trials. The monkey's reaches were earlier in easy trials than in difficult or misleading trials. There was also a trend for decisions to be made at a lower level of accuracy as time passed. Both monkeys and humans (Cisek, Puskas, & El-Murr, 2009) performing this task decreased their accuracy criterion over time. Other work suggested that this was due to a growing urgency signal (Churchland, Kiani, & Shadlen, 2008; Cisek et al., 2009; Standage, You, Wang, & Dorris, 2011). Faced with a time sensitive decision, there is an increased need to act as the deadline looms.

2 Methods

2.1 Network Architecture

We simulate the monkey experiment (Thura & Cisek, 2014) with a custom, attractor neural-network having two pools of excitatory units each tuned to a particular decision (left vs. right). Each token jump sends an excitatory signal to the pool that is tuned to a particular side. Each of the two excitatory pools, in turn, excites its own pool of inhibitory units, which send inhibitory signals to the opposing excitatory pool. These differing numbers correspond to the approximate ratio of excitatory to inhibitory brain neurons (Song, Yang, & Wang, 2016). Such attractor networks have been used to simulate both decision-making and working-memory phenomena in a variety of contexts (Grossberg, 1982; Wang, 2009).

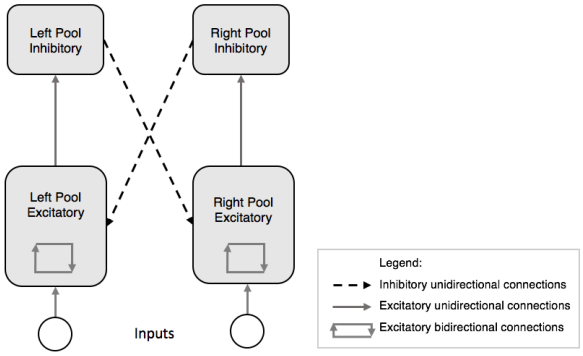


Figure 1: Network architecture

2.2 Inputs

A trial consists of a randomly generated sequence of a total of 15 left or right token jumps. A jump is simulated by presenting a 1 to the excitatory pool that is tuned to the corresponding side. We control level of difficulty by the proportions of token jumps to each side. The proportion of tokens jumping toward the correct side and incorrect side for easy, medium, and hard difficulty are 0.8 and 0.2, 0.7 and 0.3, and 0.55 and 0.45, respectively. Each time cycle is represented by one token jump.

2.3 Connection weights and activation changes

Units within each excitatory pool are fully interconnected with connection weights of .0325. The 75 excitatory units in each pool send excitatory signals to a linked pool of 25 inhibitory units to which they are fully and unidirectioanlly connected, with connection weights of .15. These inhibitory units are not inter-connected, but send inhibitory signals to the opposite excitatory pool due to weights of -.06, also unidirectionally.

At decision time, the excitatory pool with the higher activation is considered the chosen side, and the suppressed excitatory pool is the unchosen side. A unit's activation level is updated by taking its net input and applying an excitatory or inhibitory shifted sigmoidal activation function scaled by our urgency function. The excitatory activation function is shifted such that activation is at rest when net input is 0. Inhibitory unit activations stem from positive excitatory units and positive weights, and have a minimum net activation of 0. To ensure that inhibitory units can have activations ranging from 0-1, we shift the curve such that activation is also 0 when net input is 0. The urgency signal is modulated by time and the absolute difference of tokens at each of the two sides.

Each time cycle, 150 excitatory units are randomly selected, with replacement, to be updated. When an excitatory unit is updated, its 25 corresponding inhibitory units are then updated. At the end of each time cycle, the units decay according to a decay rate.

2.4 Simulations

We run our model for single decision trials and commitment tests. We analyze time to decision, activation levels, and commitments, and compare simulation results to monkey results (Thura & Cisek, 2014). To test whether commitment is reached, we fix the current value of urgency at decision time and continue the current trial. After 15 token jumps, the same network continues with another 15 jumps, using an easy trial favoring the alternate target.

3 Results

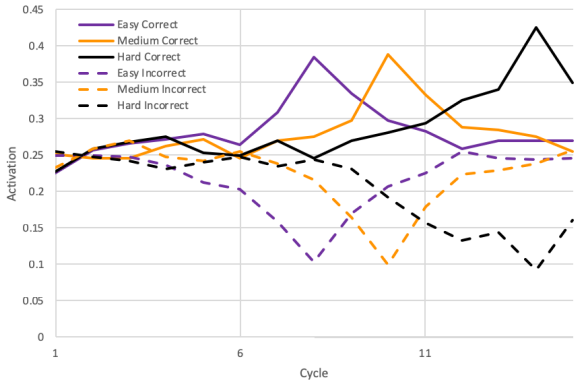
3.1 Activation patterns for single decisions

Activation patterns behave much the same at each level of difficulty. As seen in Figure 2B, activation in an excitatory pool increases when a token moves to that side, and decreases for the alternate excitatory pool**.** As seen in Figure 2A, there is a gradual buildup to an activation peak at decision, and mean peak activation for increasing difficulty is 0.39, 0.39, and 0.38 for the selected pool and 0.1 for suppressed pool. Immediately after the peak, activation quickly returns back to baseline (driven by resetting urgency). The network's activation fluctuates based on the remaining evidence. This pattern is consistent with the monkey evidence (Thura & Cisek, 2014).

3.2 Decision time and decision correctness

To determine whether the model differentiates between levels of difficulty, we test the three levels of difficulty and record the decision time and the percentage of choosing the correct target. Mean decision times (in cycles), are 6.35 for easy, 7.7 for medium, and 9.3 for hard trials. A 1-way ANOVA linear trend test is significant, $F(1, 57) = 20.4$, $p < .001$. With an LSD multiple comparison, all three means differ from each other, ps $< .05$, thus simulating the monkey data.

A



B

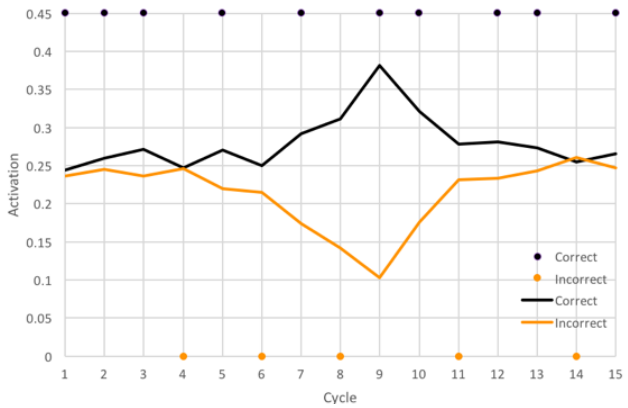


Figure 2: A) Activation levels for three single replications, illustrating the timing of activation peaks and resetting of urgency at decision. B) Example of excitatory activations in a medium trial, along with token jumps at each time cycle. Activation follows the evidence presented, and reaches a peak at decision before returning back to baseline.

Decisions are more accurate simpler trials, reproducing the trend in the monkey data. Over 20 trials, the model selects the correct target in 100% of easy trials, 95% of medium trials, and 65% of hard trials. Overall X^{2}(2) = 12.4, $p < .005$. For hard trials vs. the rest, $\mathcal{X}^{2}$(1) = 12.1, $p < .001$.

3.3 Commitment

To determine if commitment is reached at decision time, we test to see if the network falls into a stable attractor state. Activation in the selected excitatory pool rises until it reaches an asymptote, falling into a stable attractor, where contradictory new evidence over the last 15 cycles does not change the initial decision. To assess whether commitment varies with trial difficulty, we do a one-way ANOVA of last cycle activation in the winning excitatory pool with trial difficulty as the between-replication factor, $F(1, 57) = 719$, $p < .001$, $\eta^2_{partial} = .96$. All three means differ from each other by the LSD test, ps $< .001$. Although there is no comparable measure of commitment in the monkey data, this simulation result confirms the attractor characteristics of our model.

4 Discussion

Our results show that an attractor neural network with a pair of excitatory and inhibitory pools with a winner-take-all competition mediated by mutual inhibition provides a computationally sufficient way to simulate dynamic decisions in continually changing environments. Our neural attractor algorithm accurately simulates several empirical results seen in monkey experiments (Thura & Cisek, 2014): (1) an activity pattern of rising in one of two pretuned pools of excitatory neurons, (2) the correctness and latency of decisions made in various trial types, (3) approximately uniform activation levels reached in trials of varying difficulty.

There are a few additional phenomena in the monkey results that we have not yet tried to simulate, including a comparison of fast vs. slow blocks of trials, the effects of misleading evidence and analyze activation changes in the pools of inhibitory units. Analogously, the study of inhibitory neurons in neural recording experiments have not yet documented the functioning of inhibitory neurons. A study of inhibitory pools in our models could furnish useful predictions for monkey inhibitory neurons in neural recording experiments.

The current model is limited to binary decisions, whereas many realistic decisions involve more than two options. Recent eye-tracking research has found that humans decide among multiple options by making pairwise comparisons of options (Noguchi & Stewart, 2014). It could be interesting to implement such pairwise comparisons to simulate multi-option decision making. It has been proposed that there are dedicated brain circuits for processing binary decisions (Wang, 2009). Perhaps these circuits could also process multiple options using a pairwise method.

Although there are widely accepted models for simulating binary decision tasks that would produce similar results, our model offers a biologically detailed neural mechanism for making committed decisions based on continually changing evidence and confirms the role of urgency in such decisions. Additionally, this model will serve as a prediction for the role of inhibitory units in an urgency gating, attractor model. Two pools of excitatory neurons each compete through mutual inhibition. A decision is made when the excitatory activity of one pool suppresses the other excitatory pool.

5 References

Churchland, A. K., Kiani, R., & Shadlen, M. N. (2008). Decision-making with multiple alternatives. *Nature Neuroscience*, *11*(6), 693–702.

Cisek, P., Puskas, G. A., & El-Murr, S. (2009). Decisions in changing conditions: the urgency-gating model. *J. Neurosci.*, *29*, 11560–11571.

Grossberg, S. (1982). *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*. Boston, MA: Reidel.

Song, H. F., Yang, G. R., & Wang, X. J. (2016). Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLoS Computational Biology*, *12*(2), 1–30.

Standage, D., You, H., Wang, D.-H., & Dorris, M. C. (2011). Gain modulation by an urgency signal controls the speed–accuracy trade-off in a network model of a cortical decision circuit. *Frontiers in Computational Neuroscience*, *5*(February), 1–14.

Thura, D., & Cisek, P. (2014). Deliberation and commitment in the premotor and primary motor cortex during dynamic decision making. *Neuron*, *81*(6), 1401–1416.

Wang, X. J. (2009). Attractor network models. *Encyclopedia of Neuroscience*, *1*, 667–679.

Reinforcement learning for mean-field teams

Jayakumar Subramanian
Department of Electrical and Computer Engineering
McGill University
Montreal, QC H3A0G4
jayakumar.subramanian@mail.mcgill.ca

Raihan Seraj
Department of Electrical and Computer Engineering
McGill University
Montreal, QC H3A0G4
raihan.seraj@mail.mcgill.ca

Aditya Mahajan
Department of Electrical and Computer Engineering
McGill University
Montreal, QC H3A0G4
aditya.mahajan@mcgill.ca

Abstract

We develop reinforcement learning (RL) algorithms for a class of multi-agent systems called mean-field teams (MFT). Teams are multi-agent systems where agents have a common goal and receive a common reward at each time step. The team objective is to maximize the expected cumulative discounted reward over an infinite horizon. MFTs are teams with homogeneous, anonymous agents such that the agents are coupled in their dynamics and rewards through the mean-field (i.e., empirical distribution of the agents' state). In our work, we consider MFTs with a mean-field sharing information structure, i.e., each agent knows its local state and the empirical mean-field at each time step. We obtain a dynamic programming (DP) decomposition for MFTs using a decomposition approach from literature called the common information approach, which splits the decision making process into two parts. The first part is a centralized coordination rule that yields the second part, which are prescriptions to be followed by each agent based on their local information. We develop an RL approach for MFTs under the assumption of parametrized prescriptions. We consider the parameters as actions and use conventional RL algorithms to solve the DP. We illustrate the use of these algorithms through two examples based on stylized models of the demand response problem in smart grids and malware spread in networks.

Keywords: Reinforcement learning, mean-field teams, multi-agent reinforcement learning.

1 Introduction

In this paper, we look at reinforcement learning in cooperative multi-agent systems. Several algorithms for multi-agent reinforcement learning have been proposed in the literature [2–4, 9–12, 18–21]. These algorithms perform well on certain benchmark domains but there is little theoretical analysis on whether these algorithms converge to a team optimal solution. In this paper, we present a different view on multi-agent reinforcement learning. Our central thesis is that multi-agent systems for which the team optimal planning solution can be obtained by dynamic programming [13–15], it should be straightforward to translate these dynamic programs to reinforcement learning algorithms.

2 Model

Consider a multi-agent team with n agents, indexed by the set $N = \{1, \dots, n\}$. The team operates in discrete time for an infinite horizon. Let $X_t^i \in \mathcal{X}$ and $U_t^i \in \mathcal{U}$ denote the state and action of agent $i \in N$ at time t. Note that the state space $\mathcal{X}$ and action space $\mathcal{U}$ are the same for all agents. For ease of exposition, we assume that $\mathcal{X}$ and $\mathcal{U}$ are finite sets. Given a vector $x = (x^1 \dots x^n) \in \mathcal{X}^n$ of length n, let $\xi(x)$ denote the mean-field (or empirical distribution) of x, i.e., $\xi(x) = \frac{1}{n} \sum_{i \in N} \delta_{x^i}$. Let $Z_t = \xi(X_t)$ denote the mean-field of the team at time t and $\mathcal{Z}$ denote the space of space of realizations of Z_t. Note that $\mathcal{Z}$ has at most $(n+1)^{|\mathcal{X}|}$ elements. Let $(\{x_t\}_{t \geq 0}, \{u_t\}_{t \geq 0})$ denote a realization of $(\{X_t\}_{t \geq 0}, \{U_t\}_{t \geq 0})$ and let $z_t = \xi(x_t)$. We assume that the initial states of all agents are independent, i.e., $\mathbb{P}(X_0 = x_0) = \prod_{i \in N} \mathbb{P}(X_0^i = x_0^i) =: \prod_{i \in N} P_0(x_0^i)$, where P_0 denotes the initial state distribution of agents. We assume that the global state of the system evolves in a controlled Markov manner, i.e., $\mathbb{P}(X_{t+1} = x_{t+1} \mid X_{0:t} = x_{1:t}, U_{0:t} = u_{0:t}) = \mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t, U_t = u_t)$. All agents are exchangeable, so the state evolution of a generic agent depends on the states and actions of other agents only through the mean-fields of the states, i.e., for agent i:

$$\mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t, U_t = u_t) = \prod_{i \in N} \mathbb{P}(X_t^i = x_t^i, U_t^i = u_t^i, Z_t = z_t) =: \prod_{i \in N} P(x_{t+1}^i \mid x_t^i, u_t^i, z_t),$$

where P denotes the control transition matrix. Combining all of the above, we have

$$\mathbb{P}(X_{t+1} = x_{t+1} \mid X_{0:t} = x_{0:t}, U_{0:t} = u_{0:t}) = \prod_{i \in N} P(x_{t+1}^i \mid x_t^i, u_t^i, z_t). \tag{1}$$

The system has mean-field sharing information-structure, i.e., the information available to agent i is given by: $I_t^i = \{X_t^i, Z_t\}$. We assume that all agents use identical (stochastic) control law: $\mu_t \colon \mathcal{X} \times \mathcal{Z} \to \Delta(\mathcal{U})$ to choose the control action at time t, i.e., $U_t^i \sim \mu_t(X_t^i, Z_t)$. Let $\mu = (\mu_1, \mu_2, \dots)$ denote the team policy for all times. Note that, in general, restricting attention to identical policies may lead to a loss of optimality. See [1] for an example. Nonetheless, identical policies are attractive for reasons of fairness, simplicity, and robustness.

The team receives a per-step reward given by: $R_t \sim r(\boldsymbol{X}_t, \boldsymbol{U}_t)$. Given strategy $\mu = (\mu_1, \mu_2, \dots)$ the expected total reward incurred by the team is given by:

$$J(\mu) = \mathbb{E}^{\mu} \Big[\sum_{t=0}^{\infty} \gamma^t R_t \Big], \tag{2}$$

where $\gamma \in (0, 1)$ is the discount factor. The objective is to choose a policy μ to maximize the performance $J(\mu)$ given by (2).

3 Solution approach

The mean-field team model formulated above is a multi-agent team problem with non classical information structure. A planning solution of this model was presented in [1], which we summarize below for completeness. We then present a framework for using reinforcement learning in such models.

3.1 Planning solution for mean-field teams

Given any policy $\mu = (\mu_1, \mu_2, \dots)$ and any realization, $z = (z_1, z_2, \dots)$ of the mean-field, define *prescriptions* $h_t : \mathcal{X} \to \Delta(A)$ given by $h_t(x) = \mu_t(x, z_t), \quad \forall x \in \mathcal{X}$. Let $\mathcal{H}$ denote the space of all such prescritions. When the mean field trajectory is a random process, the prescriptions h_t is a random vector which we denote H_t. The results of [1] relies on the following two key properties. Let $(z_{1:t+1}, h_{1:t})$ denote any realization of $(Z_{1:t+1}, H_{1:t})$. We have:

1. $\{Z_t\}_{t \geq 1}$ is a controlled Markov process with control action h_t, i.e., $\mathbb{P}^{\mu}(Z_{t+1} = z_{t+1} \mid Z_{1:t} = z_{1:t}, H_{1:t} = h_{1:t}) = \mathbb{P}(Z_{t+1} = z_{t+1} \mid Z_t = z_t, H_t = h_t)$. Note that the right hand side does not depend on the choice of decision rule μ. Furthermore, the right hand side can be simplified as: $\mathbb{P}(Z_{t+1} = z_t \mid Z_t = z_t, H_t = h_t) = \sum_{x_{t+1} : \xi(x_{t+1}) = z_{t+1}} \prod_{i \in N} P(x_{t+1}^i \mid x_t^i, h_t(x_t^i), z_t)$, where x_t is any state such that $\xi(x_t) = z_t$.

2. The expected per-step reward simplifies as follows. $\mathbb{E}[r(\boldsymbol{X}_t, \boldsymbol{U}_t)|Z_{1:t}, H_{1:t}] = \mathbb{E}[r(\boldsymbol{X}_t, \boldsymbol{U}_t)|Z_t, H_t] =: \tilde{r}(Z_t, H_t)$.

It is shown in [1] that these two properties imply that the optimal policy μ can be identified as follows.

Theorem 1 *Let $V : \mathcal{Z} \to \mathbb{R}$ be the unique bounded fixed point of the following equation:*

$$V(z) := \max_{h \in \mathcal{H}} \mathbb{E}[\tilde{r}(z, h) + \gamma V(Z_{t+1})|Z_t = z, H_t = h]. \tag{3}$$

Let $\psi(z)$ be an $\arg\max$ *of the right hand side of* (3)*. Then the policy, $\mu(x, z) = \psi(z)(x)$, is an optimal policy for Problem* (2)*.* □

The action space $\mathcal{H}$ of the above dynamic program is all functions functions from $\mathcal{X}$ to $\Delta(\mathcal{U})$. We assume that $\mathcal{H}$ is approximated by some family of parametrized functions $\mathcal{H}_\Phi = \{h_\phi\}_{\phi \in \Phi}$ (where Φ is a compact and convex set) such as Gibbs/Boltzmann functions or neural networks. With such a parametrization, the dynamic program of (3) may be approximated as:

$$V(z) = \max_{\phi \in \Phi} \mathbb{E}[\tilde{r}(z, h_\phi) + \gamma V(Z_{t+1}) \mid Z_t = z, H_t = h_\phi] \tag{4}$$

Let $\hat{\psi}(z)$ be an $\arg\max$ of the right hand side of (4). Then the policy, $\mu(x, z) = h_{\hat{\psi}(z)}(x)$, is the best policy for Problem (2) when $\mu_t(\cdot, z_t)$ is restricted to belong to $\mathcal{H}_\Phi$.

3.2 Reinforcement learning for mean-field teams (MFT-RL)

In this section, we present a reinforcement learning algorithm for the special case where the reward is a cumulative reward, i.e., $R_t = \frac{1}{n}\sum_{i \in N} R_t^i$, where $R_t^i \sim \hat{r}(X_t^i, U_t^i, Z_t)$. We assume that we have access to a simulator for $P(\cdot \mid x_t^i, u_t^i, z_t)$ and $\hat{r}(x_t^i, u_t^i, z_t)$. This simulator is for a generic agent and takes the current local state, current local action and current mean-field as input and generates a sample of the local next state and the total reward as output. Using n copies of this simulator, we create a simulator for the mean-field dynamics. We start with n agents with initial local state sampled according to P_0. We assume that all these agents use a common stochastic policy $\hat{\psi} : \mathcal{Z} \to \Phi$ to generate prescription parameters $\phi_t \sim \hat{\psi}(z_t)$. Given this sampled value of ϕ_t, each agent independently samples a control action $u_t^i \sim h_{\phi_t}(x_t^i)$. The actions u_t^i of agent i and the current mean-field z_t are given as input to the i^{th} simulator and the sampled output (X_{t+1}^i, R_t^i) are averaged to obtain (Z_{t+1}, R_t). Thus, we have a simulator with internal state z_t. This simulator takes ϕ_t as an input and gives (Z_{t+1}, R_t) as sampled next mean-field state and reward. Thus, this is a simulator for $P(z_{t+1} \mid z_t, h_{\phi_t})$ and $\tilde{r}(z_t, h_{\phi_t})$. We can use this simulator with any standard RL algorithm to find the optimal policy for the dynamic program (4). In our experiments below, we use TRPO [16], PPO [17] and NAFDQN [5].

4 Numerical experiments

4.1 Benchmark domains

We consider the following domains to illustrate different decentralized reinforcement learning algorithms.

4.1.1 Demand response in smart grids

This is a stylized model for demand response in smart grids [1]. The system consists of n agents, where $\mathcal{X} = \{0, 1\}$, $\mathcal{U} = \{\emptyset, 0, 1\}$,

$$P(\cdot \mid \cdot, \emptyset, z) = M \tag{5}$$
$$P(\cdot \mid \cdot, 0, z) = (1 - \varepsilon_1) \left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right] + \varepsilon_1 M \tag{6}$$
$$P(\cdot \mid \cdot, 1, z) = (1 - \varepsilon_2) \left[\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\right] + \varepsilon_2 M, \tag{7}$$

where M denotes the "natural" dynamics of the systems and ε_1 and ε_2 are small positive constants.

The per-step reward is given by: $R_t = -\left(\frac{1}{n}\sum_{i \in N}\left(c_0 \mathbb{1}_{\{U_t^i = 0\}} + c_1 \mathbb{1}_{\{U_t^i = 1\}}\right) + KL(Z_t \| \zeta)\right)$, where c_0 and c_1 are costs for taking actions 0 and 1 respectively, ζ is a given target distribution and $KL(Z_t \| \zeta)$ denotes the Kullback-Leibler divergence between Z_t and ζ. In our experiments, we consider we consider a system with $n = 100$ agents, initial state distribution $P_0 = [1/3, 2/3]$, $M = \left[\begin{smallmatrix} 0.25 & 0.75 \\ 0.375 & 0.625 \end{smallmatrix}\right]$, $c_0 = 0.1$, $c_1 = 0.2$, $\zeta = [0.7, 0.3]$, $\varepsilon_1 = \varepsilon_2 = 0.2$ and discount factor $\gamma = 0.9$.

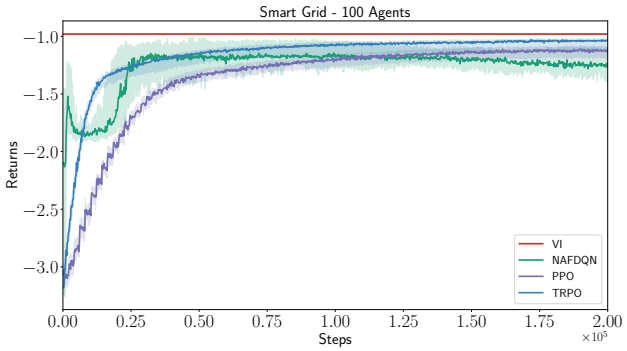


Figure 1: Demand response domain (25 independent runs).

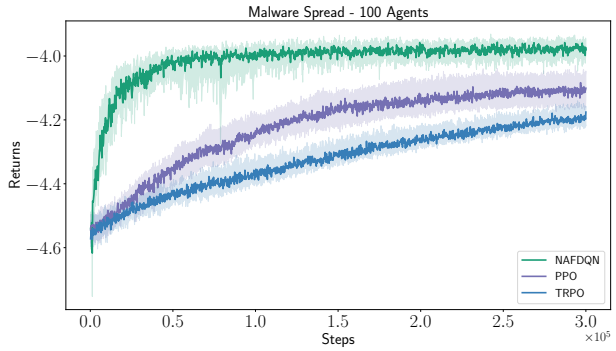


Figure 2: Malware spread domain (15 independent runs).

4.1.2 Malware spread in networks

This is a stylized model for malware spread in networks [6–8]. The system consists of n agents where $\mathcal{X} = [0, 1], \mathcal{U} = \{0, 1\}$. The dynamics are given by:

$$X^i_{t+1} = \begin{cases} X^i_t + (1 - X^i_t)\omega_t, & \text{for } U_t = 0, \\ 0 & \text{for } U_t = 1, \end{cases}$$

where $\omega_t \sim \texttt{Uniform}[0, 1]$. The per-step reward is given by: $R_t = -\left(\frac{1}{n}\sum_{i\in N}(k + \langle Z_t\rangle)X^i_t + \lambda U^i_t\right)$, where $\langle Z_t\rangle$ denotes the average of Z_t, and λ is the cost of taking action 1. In our experiments, we consider $k = 0.2$, initial state distribution $P_0 = \texttt{Uniform}(\mathcal{X})$, $\lambda = 0.5$ and discount factor $\gamma = 0.9$. For the simulation, we discretize the state space into 11 bins—$0, 0.1, \ldots, 1$.

4.2 Simulation results

We consider three variants of MFT-RL algorithms, which use different RL algorithms for the mean-field system—TRPO, PPO and NAFDQN. Figure 1 shows the result for the demand response domain and Figure 2 shows the result for the malware spread domain. For each of the MFT-RL algorithms, the dark line shows the median performance and the shaded region shows the region between the first and third quartiles across multiple independent runs. For the demand response domain we also show the optimal performance obtained using the value iteration algorithm presented in [1]. All these variants of MFT-RL algorithms converge almost to the optimal value.

4.3 Mean-field approximations

Mean-field approximations are a common approach to simplify the planning solution of mean-field coupled systems. The main idea is to approximate a large population system with an infinite population system, find the optimal policy for the infinite population system and use that policy in the finite population system. Under appropriate regularity conditions, it can be shown that such an approximate policy is ε-optimal where ε is $O(1/n)$ or $O(1/\sqrt{n})$. Such approximations rely on the system model and are not appropriate in the learning setup. However, the mean-field approximation results suggest some form of continuity in the optimal policy as the number of agents becomes large. This motivates us to investigate the reverse question. Can we find an approximate policy for a $n-$agent mean-field team by running MFT-RL on m agents, where $m < n$?

We investigate this idea in the demand response domain. We use MFT-RL for $m = 100$ agents using TRPO and PPO, and use the resultant policy in the systems with $n > 100$ agents. We compare this performance with optimal planning solution obtained using value iteration. The results are shown in Figure 3. This shows that the policy obtained for the 100 agent RL environment performs reasonably well in environments with larger number of agents as well.

5 Conclusion

There are many results in the Dec-POMDP/decentralized control literature where a team optimal solution can be obtained using dynamic programming. Our central thesis is that for such models one can easily translate the dynamic program to a reinforcement learning algorithm. We illustrate this point by using mean-field teams as an example. This allows us to use standard off-the-shelf RL algorithms to obtain solutions for some MARL setups. The numerical results show that standard single agent RL algorithms work for RL for MFTs.

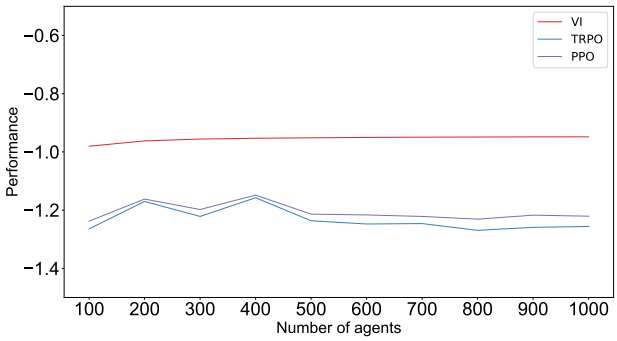


Figure 3: Performance of policy obtained in 100 agent system in systems with larger number of agents.

References

[1] ARABNEYDI, J., AND MAHAJAN, A. Team optimal control of coupled subsystems with mean-field sharing. In *IEEE CDC* (2014), pp. 1669–1674.

[2] BUŞONIU, L., BABUŠKA, R., AND DE SCHUTTER, B. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*. Springer, 2010, pp. 183–221.

[3] FOERSTER, J., NARDELLI, N., FARQUHAR, G., TORR, P., KOHLI, P., WHITESON, S., ET AL. Stabilising experience replay for deep multi-agent reinforcement learning. arXiv:1702.08887, 2017.

[4] FOERSTER, J. N., SONG, F., HUGHES, E., BURCH, N., DUNNING, I., WHITESON, S., BOTVINICK, M., AND BOWLING, M. Bayesian action decoder for deep multi-agent reinforcement learning. arXiv:1811.01458, 2018.

[5] GU, S., LILLICRAP, T., SUTSKEVER, I., AND LEVINE, S. Continuous deep Q-learning with model-based acceleration. In *ICML* (2016).

[6] HUANG, M., AND MA, Y. Mean field stochastic games: Monotone costs and threshold policies. In *IEEE CDC* (2016), pp. 7105–7110.

[7] HUANG, M., AND MA, Y. Mean field stochastic games with binary action spaces and monotone costs. arXiv:1701.06661, 2017.

[8] HUANG, M., AND MA, Y. Mean field stochastic games with binary actions: Stationary threshold policies. In *IEEE CDC* (2017), pp. 27–32.

[9] LEIBO, J. Z., ZAMBALDI, V., LANCTOT, M., MARECKI, J., AND GRAEPEL, T. Multi-agent reinforcement learning in sequential social dilemmas. In *AAMAS* (2017).

[10] LITTMAN, M. L. Markov games as a framework for multi-agent reinforcement learning. In *ICML.* (1994).

[11] LITTMAN, M. L. Friend-or-foe q-learning in general-sum games. In *ICML.* (2001).

[12] LITTMAN, M. L. Value-function reinforcement learning in markov games. *Cogn. Sys. Research 2*, 1 (2001), 55–66.

[13] MAHAJAN, A., AND MANNAN, M. Decentralized stochastic control. *Ann Oper Res., 241* (June 2016), 109–126.

[14] NAYYAR, A., MAHAJAN, A., AND TENEKETZIS, D. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE TAC 58*, 7 (2013), 1644–1658.

[15] OLIEHOEK, F. A., AND AMATO, C. *A concise introduction to decentralized POMDPs*, vol. 1. Springer, 2015.

[16] SCHULMAN, J., LEVINE, S., ABBEEL, P., JORDAN, M., AND MORITZ, P. Trust region policy optimization. In *ICML* (2015).

[17] SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A., AND KLIMOV, O. Proximal policy optimization algorithms. arXiv:1707.06347, 2017.

[18] YANG, J., YE, X., TRIVEDI, R., XU, H., AND ZHA, H. Deep mean field games for learning optimal behavior policy of large populations. In *ICLR* (2018).

[19] YANG, Y., LUO, R., LI, M., ZHOU, M., ZHANG, W., AND WANG, J. Mean field multi-agent reinforcement learning. In *ICML* (2018).

[20] YIN, H., MEHTA, P. G., MEYN, S. P., AND SHANBHAG, U. V. Learning in mean-field games. *IEEE TAC* (Mar 2014), 629–644.

[21] ZHANG, K., YANG, Z., LIU, H., ZHANG, T., AND BASAR, T. Fully decentralized multi-agent reinforcement learning with networked agents. In *ICML* (2018).

Approximate information state for partially observed systems

Jayakumar Subramanian
Electrical and Computer Engineering
McGill University, Montreal, QC H2T2A7
jayakumar.subramanian@mail.mcgill.ca

Aditya Mahajan
Electrical and Computer Engineering
McGill University, Montreal, QC H2T2A7
aditya.mahajan@mcgill.ca

Abstract

The standard approach for modeling partially observed systems is to model them as partially observable Markov decision processes (POMDPs) and obtain a dynamic program in terms of a belief state. The belief state formulation works well for planning but is not ideal for learning because the belief state depends on the model and, as such, is not observable when the model is unknown.

In this paper, we present an alternative notion of an information state for obtaining a dynamic program in partially observed models. In particular, an information state is a sufficient statistic for the current reward which evolves in a controlled Markov manner. We show that such an information state leads to a dynamic programming decomposition. Then we present a notion of an approximate information state and present an approximate dynamic program based on the approximate information state. Approximate information state is defined in terms of properties that can be estimated using sampled trajectories. Therefore, they provide a constructive method for reinforcement learning in partially observed systems. We present one such construction and show that it performs better than the state of the art for three benchmark models.

Keywords: Partially observable Markov decision processes; reinforcement learning; planning; approximate information state.

1 Introduction

The theory of Markov decision processes focuses primarily on systems with full state observation. When systems with partial state observations are considered, they are converted to systems with full state observations by considering the belief state (which is the posterior belief on the state of the system given the history of observations and actions). Although this leads to an explosion in the size of the state space, the resulting value function has a nice property—it is piecewise linear and convex in the belief state [14]—which is exploited to develop efficient algorithms to compute the optimal policy [9, 13]. Thus, for planning, there is little value in studying alternative characterizations of partially observed models.

However, the belief state formulation is not as nice a fit for learning. Part of the difficulty is that the construction of the belief state depends on the system model. So, when the system model is unknown, the belief state cannot be constructed using the observations. Therefore, critic based methods are not directly applicable. There are some results that circumvent this difficulty [3, 7, 10]. However, many of the recent results suggest that using RNNs (Recurrent Neural Networks [12]) or LSTMs (Long Short Term Memories [8]) for modeling the policy function (actor) and/or the action-value function (critic) works for reinforcement learning in partially observed systems [1, 2, 5, 6, 16, 17]. In this paper, we present a rigorous theory for planning and learning in partially observed models using the notions of information state and approximate information state. We then present numerical experiments that show that the approximate information state based works well on benchmark models.

2 Model

A general system with partial observations may be represented using the following stochastic input-output model. Consider a system that takes two inputs: a control input $U_t \in \mathcal{U}$ and a stochastic input $W_t \in W$ and generates two outputs: an observation $Y_t \in \mathcal{Y}$ and a real-valued reward R_t. The spaces $\mathcal{W}, \mathcal{U}$, and $\mathcal{Y}$ are Banach spaces and the stochastic inputs $(W_1, \dots, W_T)$ are independent random variables defined on a common probability space.

Formally, we assume that there are observation functions $\{f_t\}_{t=1}^T$ and reward functions $\{r_t\}_{t=1}^T$ such that $Y_t = f_t(Y_{1:t}, U_{1:t-1}, W_t)$ and $R_t = r_t(Y_{1:t}, U_{1:t}, W_t)$. An agent observes the history $H_t = (Y_{1:t}, U_{1:t-1})$ of observations and control inputs until time t and chooses the control input $U_t = \pi_t(H_t)$ according to some history dependent policy $\pi \coloneqq \{\pi_t\}_{t=1}^T$. The performance of policy π is given by

$$J(\pi) = \mathbb{E}^\pi \left[\sum_{t=1}^T R_t \right]. \tag{1}$$

The objective of the agent is to choose a policy π to maximize the expected total reward $J(\pi)$.

A dynamic programming decomposition. Recursively define the following *value functions*. $V_{T+1}(h_{T+1}) \coloneqq 0$ and for $t \in \{T, \dots, 1\}$:

$$Q_t(h_t, u_t) = \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, U_t = u_t] \quad \text{and} \quad V_t(h_t) = \max_{u_t \in \mathcal{U}} Q_t(h_t, u_t). \tag{2}$$

Theorem 1 *A policy $\pi = (\pi_1, \dots, \pi_T)$ is optimal if and only if it satisfies $\pi_t(h_t) \in \arg\max_{u_t \in \mathcal{U}} Q_t(h_t, u_t)$.*

The above dynamic program uses the history of observations and actions as state and as such a dynamic program is not efficient for computing the optimal policy but it will serve as a reference for the rest of the analysis.

Information state and a simplified dynamic program. Let $\mathcal{F}_t = \sigma(H_t)$ denote the filtration generated by the history of observations and control actions.

Definition 1 An information state $\{Z_t\}_{t \geq 1}$, $Z_t \in \mathcal{Z}$, is an $\mathcal{F}_t$ adapted process (therefore, there exist functions $\{\vartheta_t\}_{t=1}^T$ such that $Z_t = \vartheta_t(H_t)$) that satisfies the following properties:

(P1) **Sufficient for performance evaluation**, i.e., $\mathbb{E}[R_t \mid H_t = h_t, U_t = u_t] = \mathbb{E}[R_t \mid Z_t = \vartheta_t(h_t), U_t = u_t]$.

(P2) **Sufficient to predict itself**, i.e., $\mathbb{P}(Z_{t+1} = z_{t+1} | H_t = h_t, U_t = u_t) = \mathbb{P}(Z_{t+1} = z_{t+1} | Z_t = z_t, U_t = u_t)$, for all z_{t+1}.

There is no restriction on the space $\mathcal{Z}$, although an information state is useful only when the space $\mathcal{Z}$ is "small" in an appropriate sense. We have assumed that the space $\mathcal{Z}$ is time-homogeneous for convenience. In some situations, it may be more convenient to construct an information state which takes values in spaces that are changing with time.

For some models, instead of (P2), it is easier to verify the following stronger conditions:

(P2a) **Evolves in a state-like manner**, i.e., there exist measurable functions $\{\varphi_t\}_{t=1}^T$ such that $Z_{t+1} = \varphi_t(Z_t, Y_{t+1}, U_t)$.

(P2b) Is sufficient for predicting future observations, i.e., for any y_{t+1}

$$\mathbb{P}(Y_{t+1} = y_{t+1} \mid H_t = h_t, U_t = u_t) = \mathbb{P}(Y_{t+1} = y_{t+1} \mid Z_t = \vartheta_t(h_t), U_t = u_t).$$

Proposition 1 *(P2a) and (P2b) imply (P2).*

Note that $Z_t = H_t$ is always an information state, so an information state always exists. It is straight-forward to show that if we construct a state space model for the above input-output model, then the belief on the state given the history of observations and controls is an information state. Below we present an example of a non-trivial information state that is much simpler than the belief state.

Example 1 (Machine Maintenance) Consider a machine which can be in one of n ordered states where the first state is the best and the last state is the worst. The production cost increases with the state of the machine. The state evolves in a Markovian manner. At each time, an agent has the option to either run the machine or stop and inspect it for a cost. After inspection, s/he may either repair it (at a cost that depends on the state) or replace it (at a fixed cost). The objective is to identify a maintenance policy to determine to minimize the cost of production, inspection, repair, and replacement.

Let τ denote the time of last inspection and S_τ denote the state of the machine after inspection, repair, or replacement. Then, it can be shown that $(S_\tau, t-\tau)$ is an information state for the system.

The main feature of an information state is that one can always write a dynamic program based on an information state.

Theorem 2 *Let $\{Z_t\}_{t=1}^T$ be an information state. Recursively define value functions $\{\tilde{V}_t\}_{t=1}^{T+1}$, where $\tilde{V}_t \colon Z_t \mapsto \mathbb{R}$ as follows: $\tilde{V}_{T+1}(z_{T+1}) = 0$ and for $t \in \{T, \dots, 1\}$:*

$$\tilde{Q}_t(z_t, u_t) = \mathbb{E}[R_t + \tilde{V}_{t+1}(Z_{t+1}) \mid Z_t = z_t, U_t = u_t] \quad \textit{and} \quad \tilde{V}_t(z_t) = \max_{u_t \in \mathcal{U}} Q_t(z_t, u_t). \tag{3}$$

Then, $Q_t(h_t, u_t) = \tilde{Q}_t(\vartheta_t(h_t), u_t)$ and $V_t(h_t) = \tilde{V}_t(\vartheta_t(h_t))$.

Remark 1 In light of Theorem 2, an information state may be viewed as a generalization of the traditional notion of state [11, 18]. Traditionally, the state of an input-output system is sufficient for input-output mapping. In contrast, the information state is sufficient for dynamic programming.

The notion of information state is also related to sufficient statistics for optimal control [15]. However, in contrast to [15], we do not assume a state space model for the underlying system so it is easier to develop reinforcement learning algorithms using our notion of an information state.

Coming back to Example 1, Theorem 2 shows that we can write a dynamic program for that model using the information state $(S_\tau, t-\tau)$, which takes values in a countable set. This countable state dynamic program is considerably simpler than the standard belief state dynamic program typically used for that model. Another feature of the information state formulation is that the information state $(S_\tau, t-\tau)$ does not depend on the transition probability of the state of the machine or the cost of inspection or repair. Thus, if these model parameters were unknown, we can use a standard reinforcement learning algorithm to find an optimal policy which maps $(S_\tau, t-\tau)$ to current action.

Given these benefits of a good information state, it is natural to consider a data-driven approach to identify an information state. An information state identified from data will not be exact and it is important to understand what is the loss in performance when using an approximate information state. In the next section, we present a notion of approximate information state and bound the approximation error.

3 Approximate information state (AIS)

Roughly speaking, a compression of the history is an approximate information state if it approximately satisfies (P1) and (P2). This intuition can be made precise as follows.

Definition 2 Given positive numbers ε and δ, an (ε, δ)-approximate information state $\{\hat{Z}_t\}_{t=1}^T$, where $\hat{Z}_t$ takes values in a in a Polish metric space $(\hat{\mathcal{Z}}, d)$, is an $\mathcal{F}_t$ adapted process (therefore, there exist functions $\{\widehat{\vartheta}_t\}_{t=1}^T$ such that $\hat{Z}_t = \widehat{\vartheta}_t(H_t)$) that satisfies the following properties:

(AP1) Sufficient for approx. performance evaluation, i.e., $\big|\mathbb{E}[R_t \mid H_t = h_t, U_t = u_t] - \mathbb{E}[R_t \mid \hat{Z}_t = \widehat{\vartheta}_t(h_t), U_t = u_t]\big| \le \varepsilon$.

(AP2) Sufficient to predict itself approximately. For any Borel subset A of $\hat{\mathcal{Z}}$ define, $\mu_t(A) = \mathbb{P}(\hat{Z}_{t+1} \in A \mid H_t = h_t, U_t = u_t)$ and $\nu_t(A) = \mathbb{P}(\hat{Z}_{t+1} \in A \mid \hat{Z}_t = \widehat{\vartheta}_t(h_t), U_t = u_t)$. Then, $\mathcal{K}_d(\mu_t, \nu_t) \le \delta$, where $\mathcal{K}_d(\cdot, \cdot)$ denotes the Wasserstein or Kantorovich-Rubinstein distance[1] between two distributions.

[1] Let $(\mathcal{X}, d)$ be a Polish metric space. For any two probability measures μ, ν on $\mathcal{X}$, the Wasserstein distance between μ and ν is: $\mathcal{K}_d(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}} d(x,y)^p d\pi(x,y)$ where Π represents the product space of the two distributions.

Our main result is that one can write an approximate dynamic program based on an approximate information state.

Theorem 3 *Let $\{\hat{Z}_t\}_{t=1}^T$ be an (ε, δ)-approximate information state. Recursively define value functions $\{\widehat{V}_t\}_{t=1}^{T+1}$, where $\widehat{V}_t \colon \hat{Z}_t \mapsto \mathbb{R}$ as follows: $\widehat{V}_{T+1}(\hat{z}_{T+1}) = 0$ and for $t \in \{T, \dots, 1\}$:*

$$\widehat{Q}_t(\hat{z}_t, u_t) = \mathbb{E}[R_t + \widehat{V}_{t+1}(\hat{Z}_{t+1}) \mid \widehat{Z}_t = \hat{z}_t, U_t = u_t] \quad \text{and} \quad \widehat{V}_t(\hat{z}_t) = \max_{u_t \in \mathcal{U}} \widehat{Q}_t(\hat{z}_t, u_t).$$

Suppose $\widehat{V}_t$ is Lipschitz continuous with Lipschitz constant L_V. Then, we have the following:

$$|Q_t(h_t, u_t) - \widehat{Q}_t(\widehat{\vartheta}_t(h_t), u_t)| \le (T-t)(\varepsilon + L_V \delta) + \varepsilon \quad \text{and} \quad |V_t(h_t) = \widehat{V}_t(\widehat{\vartheta}_t(h_t))| \le (T-t)(\varepsilon + L_V \delta) + \varepsilon.$$

Based on Prop. 1, we provide an alternative characterization of an approximate information state. We can replace (AP2) with the following stronger conditions:

(AP2a) Evolves in a state-like manner, i.e., there exist measurable functions $\{\hat{\varphi}_t\}_{t=1}^T$ such that $\hat{Z}_{t+1} = \hat{\varphi}_t(\hat{Z}_t, Y_{t+1}, U_t)$. Moreover, these functions are Lipschitz in Y with Lipschitz constant L_U.

(AP2b) Is sufficient for predicting future observations approximately. For any Borel subset A of $\mathcal{Y}$ define, $\mu_t(A) = \mathbb{P}(Y_{t+1} \in A \mid H_t = h_t, U_t = u_t)$ and $\nu_t(A) = \mathbb{P}(Y_{t+1} \in A \mid \hat{Z}_t = \widehat{\vartheta}_t(h_t), U_t = u_t)$. Then, $\mathcal{K}(\mu_t, \nu_t) \le \delta$,

Proposition 2 *If (AP2) is replaced by (AP2a) and (AP2b), the result of Theorem 3 holds with L_V replaced by $L_U L_V$.*

Corollary 1 *Suppose $\{Z_t\}_{t=1}^T$ is an information state and $\{\hat{Z}_t\}_{t=1}^T$ is an (ε, δ)-approximate information state. Then for any realization h_t of H_t, we have the following:*

$$|Q_t(\vartheta_t(h_t), u_t) - \widehat{Q}_t(\widehat{\vartheta}_t(h_t), u_t)| \le (T-t)(\varepsilon + L_V \delta) + \varepsilon \quad \text{and} \quad |V_t(\vartheta_t(h_t)) - \widehat{V}_t(\widehat{\vartheta}_t(h_t))| \le (T-t)(\varepsilon + L_V \delta) + \varepsilon.$$

4 Reinforcement learning using approximate information state

In this section, we use an approximate information state to design reinforcement learning algorithms for infinite horizon POMDPs. We split our approach into two steps—a data-driven approach to construct an approximate information state and reinforcement learning using this approximate information state.

Constructing an approximate information state. The definition of approximate information state suggests two ways to construct an information state from data: either use $\hat{\vartheta}(h_t)$ to determine an approximate information state that satisfies conditions (AP1) and (AP2) or conditions (AP1), (AP2a), and (AP2b). We present the second approach here.

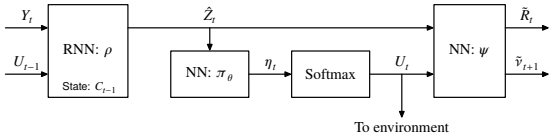
We use two function approximators: (i) A recurrent neural network (RNN) or its refinements such as LSTM or GRU with state C_{t-1}, inputs (Y_t, U_{t-1}) and output $\hat{Z}_t$. We denote this function approximator by ρ. (ii) A feed forward network as in the previous case, except that $\tilde{\nu}_{t+1}$ is the prediction of ν_{t+1}, the distribution of the next approximate information state $\hat{Z}_{t+1}$. We denote this function approximator as ψ.

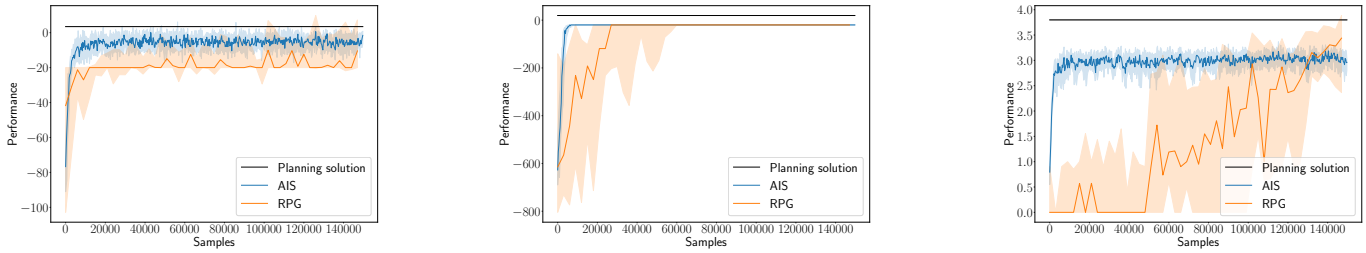
To minimize ε and δ, we train the networks ρ and ψ using the loss function $\mathcal{L}_{\rho,\psi} = \lambda \mathcal{L}_R + (1-\lambda)\mathcal{L}_\nu$ where $\mathcal{L}_R = \frac{1}{B}\sum_{t=1}^B \texttt{smoothL1}(\tilde{R}_t - R_t)$, (where B is the batch size and `smoothL1` is the standard smooth approximation for L1 loss) and $\mathcal{L}_\nu = -\sum_{t=1}^{B-1} \log(\tilde{\nu}_{t+1}(Y_{t+1}))$, which is the negative log likelihood loss for $\tilde{\nu}_t$ and thus approximates the KL-divergence between μ_t and ν_t. We use the KL-divergence as a surrogate for the Wasserstein distance because: (i) Wasserstein distance is computationally expensive to compute; and (ii) KL-divergence upper bounds the total variation (due to Pinsker's inequality), which in turn upper bounds Wasserstein distance for metric spaces with bounded diameter.

Reinforcement learning. Let $\pi_\theta : \hat{Z}_t \mapsto \Delta(U_t)$ be a parametrized stochastic policy, where the parameters θ lie in a closed convex set Θ. For example, π_θ could be a feed forward neural network with input $\hat{Z}_t$ and output to be a $|U_t|$ dimensional vector η, where: $\pi_\theta(u|\hat{z}) = \exp(\tau\eta_u)/\sum_{w\in\mathcal{U}} \exp(\tau\eta_w)$, where τ is a hyperparameter. In such a policy, θ corresponds to the weights of the network. The basic idea behind policy based reinforcement learning is to get sample path based estimates of the performance gradient $\nabla_\theta J$, which is then used as a gradient loss function for updating the parameters θ using stochastic gradient descent.

An architecture for combining the construction of the approximate information state with reinforcement learning is shown on the right. In this architecture, we train the networks (ρ, ϕ) and π_θ in parallel using a two time-scale algorithm. In particular, by a slight abuse of notation, let ρ and ψ denote the weights of the corresponding networks. Then,

$$\begin{bmatrix}\rho_{k+1}\\ \psi_{k+1}\end{bmatrix} = \begin{bmatrix}\rho_k\\ \psi_k\end{bmatrix} + a_k \nabla_{\rho,\psi}\mathcal{L}_{\rho,\psi} \text{ and } \theta_{k+1} = \theta_k + b_k \nabla_\theta J(\pi_{\theta_k}),$$





(a) Voicemail problem (b) Tiger problem (c) 4×4 grid problem

Figure 1: Performance versus samples for all examples. The solid line shows the median value and the shaded region shows the region between the first and third quartiles over 25 runs.

where the learning rates $\{a_k\}_{k\geq 1}$ and $\{b_k\}_{k\geq 1}$ satisfy the standard two time-scale stochastic approximation conditions [4].

The results of the experiment for three small dimensional POMDP benchmarks—voicemail, tiger, and 4×4 grid—are shown in Fig. 1.

References

[1] A. Baisero and C. Amato. Learning internal state models in partially observable environments;. *Reinforcement Learning under Partial Observability, NeurIPS Workshop*, 2018.

[2] B. Bakker. Reinforcement learning with long short-term memory. In *NIPS*, 2002.

[3] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *JAIR*, 15:319–350, 2001.

[4] V. S. Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294, 1997.

[5] M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. In *2015 AAAI Fall Symposium Series*, 2015.

[6] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver. Memory-based control with recurrent neural networks. *arXiv:1512.04455*, 2015.

[7] A. Hefny, Z. Marinho, W. Sun, S. Srinivasa, and G. Gordon. Recurrent predictive state policy networks. *arXiv:1803.01489*, 2018.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[10] M. L. Littman, R. S. Sutton, and S. P. Singh. Predictive representations of state. In *NIPS*, 2002.

[11] A. Nerode. Linear automaton transformations. *Proceedings of American Mathematical Society*, 9:541–544, 1958.

[12] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[13] G. Shani, J. Pineau, and R. Kaplow. A survey of point-based POMDP solvers. *AAMAS*, 2013.

[14] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

[15] C. Striebel. Sufficient statistics in the optimal control of stochastic systems. *Journal of Mathematical Analysis and Applications*, 12:576–592, 1965.

[16] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber. Solving deep memory POMDPs with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, 2007.

[17] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of the IGPL*, 18(5):620–634, 2010.

[18] H. S. Witsenhausen. Some remarks on the concept of state. In Y. C. Ho and S. K. Mitter, editors, *Directions in Large-Scale Systems*, pages 69–75. Plenum, 1976.

Validation of cognitive bias represented by reinforcement learning with asymmetric value updates

Michiyo Sugawara
Department of Cognitive and Psychological Sciences
Nagoya University
Nagoya, Aichi 4648601, Japan
sugawara.michiyo@j.mbox.nagoya-u.ac.jp

Kentaro Katahira
Department of Cognitive and Psychological Sciences
Nagoya University
Nagoya, Aichi 4648601, Japan
katahira.kentaro@b.mbox.nagoya-u.ac.jp

Abstract

Reinforcement learning (RL) models, which update the value related to a specific behavior according to a reward prediction error, have been used to model choice behavior in organisms. Recently, the magnitude of the learning rate has been reported to be biased depending on the sign of the reward prediction error. Previous studies concluded that these asymmetric learning rates reflect positivity and confirmation biases. However, Katahira (2018) reported that the tendency to repeat the same choice (perseverance) leads to pseudo asymmetric learning rates. Therefore, this study aimed to clarify whether asymmetric learning rates are the result of cognitive bias, perseverance, or both by reanalyzing the data of a previous study (Palminteri, Lefebvre, Kilford, & Blakemore, 2017). The data from the previous study consisted of two types of learning: factual and counterfactual. In the factual learning task, participants were shown the outcome of only the chosen option. By contrast, in the counterfactual learning task, participants were shown the outcomes of both the chosen and the forgone options. To accomplish the purpose of this study, we evaluated multiple RL models, including asymmetric learning rate models, perseverance models, and hybrid models. For factual learning, the asymmetric learning rate model showed that the positive learning rate was higher than the negative one, confirming the presence of positivity bias. A hybrid model incorporating the perseverance factor into the asymmetric learning rate model significantly reduced the difference between the positive and negative learning rates. In contrast to factual learning, the hybrid model did not affect the difference between positive and negative learning rates in counterfactual learning. Previous studies suggested that these biases are common in learning systems, but on the basis of these results, it is possible that different factors were present in factual and counterfactual learning (such as ambiguity).

Keywords: reinforcement learning; asymmetric learning rates; confirmation bias; positivity bias; perseverance

Acknowledgements

This study was supported by Grant-in-Aid for Scientific Research (B) (18KT0021 to KK).

1 Introduction

Reinforcement learning (RL) models have been broadly used in modeling the choice behavior of humans and animals (Daw, Gershman, Seymour, Dayan, & Dolan, 2011; Redish & Johonson, 2008). Standard RL models suppose that agents learn action-outcome associations from obtained outcomes on a trial-and-error basis (Barto, 1997). The learned action values are assumed to be updated according to the reward prediction error, which is the difference between the actual and the expected rewards (Rescorla & Wagner, 1972; Sutton & Barto, 1998). Recent studies have noted that the magnitude of an update is biased depending on the sign of the prediction error (Frank, Moustafa, Haughey, Curran, & Hutchison, 2007; Gershman, 2015, 2016; Niv, Edlund, Dayan, & O'Doherty, 2012). This bias is represented in RL models by differential learning rates for positive and negative prediction errors. Lefebvre, Lebreton, Meyniel, Bourgeois-Gironde, & Palminteri, (2017) suggested that this learning asymmetry reflects the human positivity bias in factual learning, in which feedback is given only for the option the participant choses. Palminteri et al. (2017) suggested that the learning asymmetry represent the confirmation bias in counterfactual learning, in which feedback is given for both the chosen and forgone options.

However, Katahira (2018) suggested the possibility that the estimation of asymmetric learning rates suffered from statistical artifacts caused by model misspecification. Specifically, Katahira (2018) reported that the tendency to repeat the same choices (perseverance) leads to pseudo asymmetric learning rates. The rationale is as follows. The asymmetry in the value updates induces autocorrelation of choice (i.e., the tendency to repeat the same choice or to switch to another choice, irrespective of past outcomes): the relatively larger learning rate for positive outcomes enhances the effect of positive outcomes but diminishes the effect of negative outcomes, leading to choice repetition. If an RL model without components that can directly represent the intrinsic autocorrelation–we call this component perseverance factors–is fitted to data that possess intrinsic autocorrelation (e.g., perseveration), the model tends to represent the perseveration by asymmetric learning rates. Therefore, a statistical bias that overestimates the difference in learning rates will arise.

According to the results of Katahira (2018), there is a possibility that the previous results reporting asymmetry in learning rates are due to this statistical artifact because most RL models in such studies did not include choice autocorrelation factors. However, this possibility has not yet been examined. In this study, we examine this possibility by reanalyzing the empirical data reported in Palminteri et al. (2017) with models that include the choice autocorrelation factor. We expect that if the asymmetry in estimated learning rates disappears after incorporating the choice autocorrelation factor, then the asymmetry reported in previous studies is likely to be due to artifacts caused by model misspecification.

2 Methods

2.1 Data

We used open data (https://figshare.com/authors/_/2803402) for which the analysis has been reported in Palminteri et al. (2017). Here, we briefly explain their methods. Their study included two experiments, and each experiment involved 20 participants. The participants performed a two armed-bandit task that involved choosing between two cues that were associated with outcome probabilities. The possible outcomes were either winning or losing a point. Each experiment consisted of a total of 192 trials. In the factual learning task (Experiment 1), participants were informed about the outcome of only the chosen option. In the counterfactual learning task (Experiment 2), participants were informed about both the obtained and forgone outcomes.

2.2 Models

For data from the factual learning task (Experiment 1), we used four RL models: Q model (the standard Q-learning model, also called 'One model' in Palminteri et al., 2017), Valence model (VQ model), Perseverance model (Q-P model) and Valence with Perseverance model (VQ-P model). The Q and VQ models were used in Palminteri et al. (2017). The Q-P and VQ-P models are newly included in this study to validate the perseverance factor. In the Q model, the action value for the chosen option is updated according to $Q_c(t+1) = Q_c(t) + \alpha(R_c(t) - Q_c(t))$. The outcome of trial t is denoted by $R_c(t)$. $R_c(t) - Q_c(t)$ represents the prediction error, which is denoted as δ_c. The learning rate α determines how much the model updates the action value with the prediction error. The initial action value of each option is set to zero. For data from the factual learning task (Experiment 1), only the Q value for the chosen option is updated because participants are informed about the outcome of only the chosen option. Choice probability $P_c(t)$ is determined by softmax function $P_c(t) = 1/\left[1 + \exp(-\beta(Q_c(t) - Q_u(t)))\right]$. Q_c is the Q value for the chosen option, and Q_u is the value for the unchosen option. β, called as the inverse temperature parameter, determines the sensitivity of the choice probabilities to the difference between the Q values for the two options.

The VQ model is extended from the Q model to allow for asymmetric learning rates (α_c^+, α_c^-) depending on the sign of the prediction error. Thus, the Q values are updated as follows:

$$Q_c(t+1) = \begin{cases} Q_c(t) + \alpha_c^+ \delta_c(t) & \text{if } \delta_c(t) \geq 0 \\ Q_c(t) + \alpha_c^- \delta_c(t) & \text{if } \delta_c(t) < 0 \end{cases} \quad (1)$$

In the Q-P model, the choice trace $C_i(t)$ is defined to introduce the effect of past choice into the choice probability:

$$P_c(t) = \frac{1}{1 + \exp(-\beta(Q_c(t) - Q_u(t)) - \varphi(C_c(t) - C_u(t)))} \tag{2}$$

where φ is the choice trace weight, which is a parameter that controls the tendency to repeat or avoid recently chosen options. The choice trace is computed using the following update rule:

$$C_i(t+1) = C_i(t) + \tau(I(a(t) = i) - C_i(t)) \tag{3}$$

where the indicator function $I(\cdot)$ takes on a value of 1 if the statement is true and 0 if the statement is false. The parameter τ is the decay rate of the choice trace. The VQ-P model is a hybrid of the VQ and Q-P models.

For data from the counterfactual learning task (Experiment 2), we used six RL models: Q model, Valence×Information model (VIQ model), Confirmation model (CQ model), Perseverance model (Q-P model), Valence×Information with Perseverance model (VIQ-P model), and Confirmation with Perseverance model (CQ model). The first three models were used in Palminteri et al. (2017), and the last three models were newly added in this study. Here, all models are allowed to update the Q values of both the chosen and unchosen options because participants were informed about both outcomes. The Q and Q-P models have the same parameters as in the factual learning task (Experiment 1). In these models, only one learning rate is used to update the values of both the chosen and unchosen options, regardless of the sign of the prediction error. In the VIQ model, four different learning rates are defined to represent the asymmetric updating for the chosen (α_c^+, α_c^-) and forgone (α_u^+, α_u^-) options. The Q value for the forgone option is computed as follows:

$$Q_u(t+1) = \begin{cases} Q_u(t) + \alpha_u^+ \delta_u(t) & \text{if } \delta_u(t) \geq 0 \\ Q_u(t) + \alpha_u^- \delta_u(t) & \text{if } \delta_u(t) < 0 \end{cases} \tag{4}$$

where δ_u denotes the prediction error of the forgone option. The VIQ-P model is a hybrid of the VIQ and Q-P models.

Finally, the CQ model integrates the four learning rates used in the VIQ model into two rates ($\alpha_{\text{con}}, \alpha_{\text{dis}}$) corresponding to the confirmation bias ($\alpha_{\text{con}} = \alpha_c^+ = \alpha_u^-, \alpha_{\text{dis}} = \alpha_c^- = \alpha_u^+$). The CQ-P model is a hybrid of the CQ and Q-P models.

2.3 Parameter estimation and model comparison

Parameter estimation was conducted using the maximum a posteriori method. The prior distributions and constraints followed Palminteri et al. (2017). All the learning rates were constrained to the range of $0 \leq \alpha \leq 1$ with a *Beta* (1.1, 1.1) prior distribution. The inverse temperature was constrained to the range of $0 \leq \beta \leq \infty$ with a *Gamma* (shape = 1.2, scale = 5.0) distribution. In the Perseverance model, τ was constrained to the range of $0 \leq \tau \leq 1$ with a *Beta* (1, 1) distribution, and φ was constrained to the range of $-10 \leq \varphi \leq 10$ with a *Norm* ($\mu = 0, \sigma^2 = 5$) distribution. To compare models, we used the log marginal likelihood, which was estimated by Laplace approximation (Daw, 2011).

2.4 Statistical tests

One-way repeated measures analysis of variance (rmANOVA) was conducted to compare the log marginal likelihoods of the models. We also investigated the difference between learning rates. For the VQ and VQ-P models in the factual learning task, the difference between the two learning rates was compared by using a paired t test. For the VIQ and VIQ-P models in the counterfactual learning task, two-way rmANOVAs were performed to test for differences among the four learning rates. To correct for the violation of the sphericity assumption, Greenhouse-Geiser's adjustment of the degrees of freedom was used for all rmANOVAs when appropriate. Post hoc pairwise comparisons were performed based on Shaffer's correction for multiple comparisons.

3 Results

3.1 Factual learning task

To investigate which model could best explain the data from the factual learning task, we compared four models. We found that the Q-P model had the highest log marginal likelihood. The log marginal likelihood for the models differed significantly ($F_{3,57} = 6.51, p = .018$; Table 1). Post hoc comparisons showed that three models (VQ, Q-P, and VQ-P) had higher log marginal likelihoods than the Q model ($ps < .076$). There was no difference between these three models ($ps > .129$), suggesting that the Perseverance model was comparable with the Valence model when participants chose their behavior in factual learning. We investigated how the degree of the asymmetry in RL varied with the models. The positive learning rate was significantly greater than the negative learning rate in the VQ model ($t_{19} = 2.36, p = .029$; Figure 1A), replicating Palminteri et al. (2017), but this difference was not observed in the VQ-P model ($t_{19} = .78, p = .44$; Figure 1B). Comparison of the difference between positive and negative learning rates showed that the VQ-P model significantly reduced the difference compared with VQ model ($t_{19} = 3.95, p = 8.62 \times 10^{-4}$).

Table 1: The list of models and model selection results

	Model	Learning rates	Perseverance	Inverse temperature	# of free parameters	Log marginal likelihood
Experiment 1 (Factual learning)	Q	α	-	β	2	-99.22 (5.39)
	VQ	α_c^+, α_c^-	-	β	3	-90.20 (5.84)
	Q-P	α	τ, φ	β	4	**-88.88** (6.01)
	VQ-P	α_c^+, α_c^-	τ, φ	β	5	-89.49 (5.91)
Experiment 2 (Counterfactual learning)	Q	α	-	β	2	-89.17 (5.80)
	VIQ	$\alpha_c^+, \alpha_c^-, \alpha_u^+, \alpha_u^-$	-	β	5	-76.14 (6.67)
	CQ	$\alpha_{\text{con}}, \alpha_{\text{dis}}$	-	β	3	**-75.66** (6.61)
	Q-P	α	τ, φ	β	4	-78.13 (6.28)
	VIQ-P	$\alpha_c^+, \alpha_c^-, \alpha_u^+, \alpha_u^-$	τ, φ	β	7	-77.28 (6.69)
	CQ-P	$\alpha_{\text{con}}, \alpha_{\text{dis}}$	τ, φ	β	5	-76.51 (6.59)

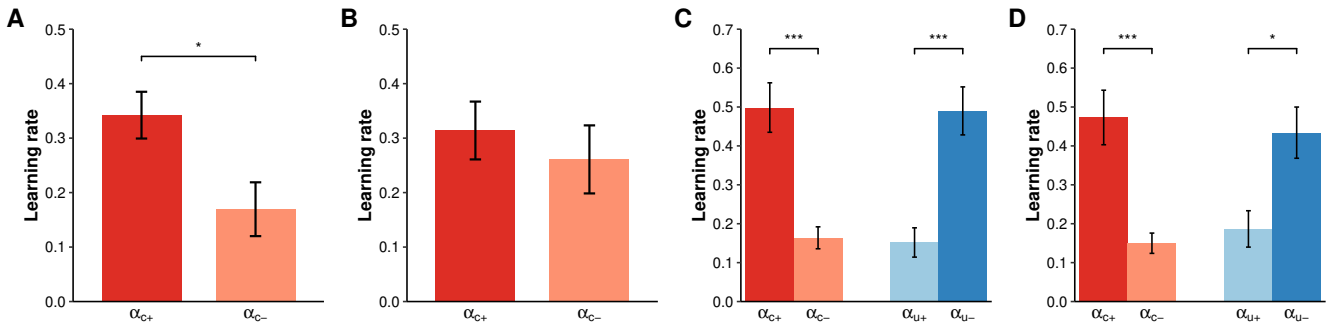


Figure 1: Learning rates for the VQ model (A) and VQ-P model (B) in the factual learning task (Palminteri et al. 2017). *$p < .05$, two-tailed paired t test. Learning rates for the VIQ model (C) and VIQ-P model (D) in the counterfactual learning task (Palminteri et al. 2017). ***$p < .001$, *$p < .05$. Post hoc analyses used the modified sequentially rejective Bonferroni procedure. All error bars represent the standard error of the mean.

3.2 Counterfactual learning task

Similar to the factual learning task, we compared six models by using the log marginal likelihood and found that the CQ model had the highest value. ANOVA showed a significant difference between models ($F_{5,95} = 30.09, p < .001$). Post hoc comparisons indicated that the log marginal likelihood for the Q model was different from that of the other models ($ps < .001$). In addition, the CQ model had a higher log marginal likelihood than the Q-P ($p = .022$) and VIQ-P ($p = .04$) models, meaning that even with the added perseverance factor, the CQ model best explained the data of the counterfactual learning task. Then, we compared the four learning rates in the VIQ model. ANOVA showed that the interaction between valence and information was significant ($F_{1,19} = 124.88, p < .001$; Figure 1C) but that the main effects of valence and information were not significant ($F_{1,19} = .04, p = .837$; $F_{1,19} = 6.0 \times 10^{-4}, p = .981$). According to the post hoc comparisons, α_c^+ and α_u^- were greater than α_c^- and α_u^+ ($ps < .001$), whereas the other pairs did not show any significant differences. Additionally, the ANOVA for the VIQ-P model indicated a significant interaction ($F_{1,19} = 37.95, p < .001$; Figure 1D) but no main effects ($F_{1,19} = 5.0 \times 10^{-4}, p = .982$; $F_{1,19} = .30, p = .594$). The four learning rates showed the same relationship as in the VIQ model; that is, α_c^+ and α_c^- were greater than α_c^- and α_u^+ ($ps < .001$). Although we directly compared the magnitude of the asymmetric learning rates ($\alpha_c^+ - \alpha_c^- + \alpha_u^- - \alpha_u^+$) in the VIQ and VIQ-P models, the difference was not significant ($t_{19} = 1.24, p = .23$). These results support the idea that the pattern of choice behavior in the counterfactual learning task reflects confirmation bias (Palminteri et al., 2017).

4 Discussion and Conclusion

This study aimed to validate the findings in Palminteri et al. (2017) about the underlying mechanisms of the asymmetric learning rates in the RL model by examining the possibility of the statistical artifact suggested in Katahira (2018).

For data from the factual learning task (Experiment 1), the results suggested that the Q-P model, which incorporated the perseverance factor into a single learning rate model, was the best model. Additionally, the asymmetric learning rate model (VQ) was comparable with the hybrid model (VQ-P). Although the VQ model had a positive learning rate that

was greater than the negative learning rate, in the VQ-P model, this asymmetry diminished. These results support the idea of Katahira (2018) that asymmetric learning rates cause an autocorrelation of choice, which can be explained by the perseverance factor. By contrast, the CQ model was the best in the counterfactual learning task (Experiment 2). Similar to the VIQ model, the VIQ-P model, which incorporates the perseverance factor into the VIQ model, showed robust asymmetric learning rates in accordance with confirmation bias. Thus, these results support the findings of Palminteri et al. (2017).

The present findings raise new questions about the effects of learning context. The results of this study suggest that the apparently asymmetric learning rates in the factual learning task (Experiment 1) can be explained by a confounding factor, that is, intrinsic perseveration. However, the asymmetric learning rates in the counterfactual learning task (Experiment 2) cannot be explained solely by the perseverance factor. Palminteri et al. (2017) suggested that both factual and counterfactual learning are different aspects of a common learning system. However, our present findings imply the existence of another factor. One possibility is the increased information in counterfactual learning. In factual learning, the outcome of the forgone option remains unclear. This ambiguity leads to decreased choice probability of that option (Hsu, Bhatt, Adolphs, Tranel, & Camerer, 2005). Ambiguity aversion leads to repeated selection of the same choice, independent of reward expectations. Because counterfactual learning provides the outcomes of both the chosen and unchosen options, it is reasonable that the effect of cognitive bias results from decreased ambiguity.

The present study found that whether perseverance explains the asymmetric learning rates depends on the learning context and that cognitive bias can explain the asymmetric learning rates in certain contexts. Taken together, both cognitive bias and perseverance are important factors to explain choice behavior, in which the dominance of these factors may depend on the available information.

References

[1] Barto, A. G. (1997). Neural Systems for Control. (O. M. Omidvar & D. L. Elliott, Eds.), Reinforcement learning.

[2] Daw, N. D. (2011). Trial-by-trial data analysis using computational models. In Decision Making, Affect, and Learning: Attention and Performance XXIII Oxford University Press, 3–38.

[3] Daw, N. D., Gershman, S. J., Seymour, B., Dayan, P., & Dolan, R. J. (2011). Model-Based Influences on Humans ' Choices and Striatal Prediction Errors. Neuron, 69(6), 1204–1215.

[4] Frank, M. J., Moustafa, A. A., Haughey, H. M., Curran, T., & Hutchison, K. E. (2007). Genetic triple dissociation reveals multiple roles for dopamine in reinforcement learning. Proceedings of the National Academy of Sciences, 104(41), 16311–16316.

[5] Gershman, S. J. (2015). Do learning rates adapt to the distribution of rewards? Psychonomic Bulletin & Review, 22(5), 1320–1327.

[6] Gershman, S. J. (2016). Empirical priors for reinforcement learning models. Journal of Mathematical Psychology, 71, 1–6.

[7] Hsu, M., Bhatt, M., Adolphs, R., Tranel, D., & Camerer, C. F. (2005). Neural systems responding to degrees of uncertainty in human decision-making. Science, 310(5754), 1680–1683.

[8] Katahira, K. (2018). The statistical structures of reinforcement learning with asymmetric value updates. Journal of Mathematical Psychology, 87, 31–45.

[9] Lefebvre, G., Lebreton, M., Meyniel, F., Bourgeois-Gironde, S., & Palminteri, S. (2017). Behavioural and neural characterization of optimistic reinforcement learning. Nature Human Behaviour, 1(4), 1–9.

[10] Niv, Y., Edlund, J., Dayan, P., & O' Doherty, J. (2012). Neural prediction errors reveal a risk-sensitive reinforcement-learning process in the human brain. Journal of Neuroscience, 32(2), 551–562.

[11] Palminteri, S. stefano palminteri's public data. https://figshare.com/authors/_/2803402

[12] Palminteri, S., Lefebvre, G., Kilford, E. J., & Blakemore, S. (2017). Confirmation bias in human reinforcement learning: Evidence from counterfactual feedback processing. PLOS Computational Biology, 13(8), e1005684.

[13] Redish, A. D., & Johnson, A. (2008). A unified framework for addiction: vulnerabilities in the decision process. Behavioral Brain Science, 31(4), 415–487.

[14] Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement. In: Classical conditioning II: current research and theory. (P. W. Black AH, Ed.).

[15] Sutton, R. S., & Barto., A. G. (1998). Reinforcement Learning: An Introduction. (C. U. Press, Ed.), Kybernetes (Vol. 27).

Inverse Reinforcement Learning from a Learning Agent

Vincent Kubala
Department of Computer Science
Brown University
Providence, RI 02912
vincent_kubala@alumni.brown.edu

George Konidaris
Department of Computer Science
Brown University
Providence, RI 02912
gdk@cs.brown.edu

Amy Greenwald
Department of Computer Science
Brown University
Providence, RI 02912
amy@cs.brown.edu

Abstract

We consider the problem of inferring the reward function and predicting the future behavior of an agent that is learning. To tackle this problem, we generalize an existing Bayesian inverse reinforcement learning algorithm to allow the actor's policy to change over time, as a function of their experiences, and to simultaneously infer the actor's reward function and methods of learning and making decisions. We show experimentally that our algorithm outperforms its traditional inverse reinforcement learning counterpart.

Keywords: Inverse reinforcement learning, Bayesian inference

1 Introduction

Inverse reinforcement learning (IRL) is the problem of inferring an actor's reward function in a Markov decision process (MDP) based on their actions in that MDP. Traditional IRL algorithms assume that the actor is an expert, in the sense that their observed actions are generated by a fixed policy that is (nearly) optimal according to their reward function. In particular, traditional IRL algorithms do not allow for the possibility that the actor is learning.

However, there are many situations in which it would be useful to infer the reward function of a *learning* actor. We might wish to learn from an agent that has not yet converged to an optimal policy. Or, we might possess knowledge of the MDP's dynamics and wish to advise a learning actor on how to reach their goals. Or, we might wish to recover the reward function or predict the future behavior of one actor who is interacting with another and learning how to do so.

We generalize Bayesian IRL to accommodate an actor whose policy may change over time. We model an actor as being characterized by a utility function, jointly determined by a reward function and an exponential discounting factor, and a behavior rule that maps the actor's utility function and experiences to a policy. We then generalize an existing IRL algorithm to simultaneously infer the actor's reward function and behavior rule, assuming that the discount factor is known. We show experimentally that our algorithm outperforms its traditional IRL counterpart when the actor is learning.

2 Background: MDPs and IRL[1]

We model the environment as a *controlled Markov process* (CMP), which is a triple $\nu = (\mathcal{S}, \mathcal{A}, \mathcal{T})$, where $\mathcal{S}$ is a finite set of possible states of the environment and $\mathcal{A}$ is a finite set of actions that the actor can execute in every state. The *transition dynamics* $\mathcal{T}$ is a collection $\{\tau(\cdot|s,a) : s \in \mathcal{S}, a \in \mathcal{A}\}$ of probability distributions on $\mathcal{S}$ that describe how the state of the environment changes between timesteps; that is, $\tau(s'|s,a)$ gives the probability that the environment will "transition" from state s at time t to a state s' at time $t+1$ if the actor takes action a. Further, $\tau(s)$ alone gives the probability that the initial state of the CMP will be s.

We observe the actor taking actions in the environment, and denote the state of the environment after the actor has taken t actions as s_t and the action that the actor chose from that state as a_t. A sequence of states and actions form a *trajectory* $z = (s_0, a_0, s_1, ...)$ from a set $\mathcal{Z}$ of possible trajectories, and we write z_t to denote the t-step prefix, $(s_0, a_0, s_1, ..., s_t)$, of a given trajectory, z. We assume that the actor has preferences over trajectories that are described by some von Neumann-Morgenstern utility function $U : \mathcal{Z} \to \mathbb{R}$. We further assume that the actor has an unknown, deterministic reward function ρ over states, and that their utility for a trajectory can be expressed as the time-discounted sum of their rewards

[1]This section follows the superb presentation of IRL by Rothkopf and Dimitrakakis [8].

for each of the states in that trajectory: $U(z) \triangleq \sum_t \gamma^t \rho(s_t)$, where $\gamma \in [0, 1]$, called the *discount factor*, defines the extent to which the actor cares about the later states in z. We assume as usual and that ρ is parameterized by some vector, θ, from a set, Θ, of possible such vectors. We use ρ_θ to denote the reward function induced by parameters θ.

The combination of a CMP and a utility function define a *Markov decision process* (MDP), $\mu = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \rho, \gamma)$, which fully specifies the actor's problem setting. The actor's problem is then to choose a *policy* such that the probability distribution over their trajectory yields maximal expected utility. A policy is a collection $\{\pi(\cdot|s) : s \in \mathcal{S}\}$ of probability distributions on $\mathcal{A}$, such that $\pi(a|s)$ gives the probability that the actor takes action a from state s. We use $\mathcal{P}_\nu$ to denote the set of valid policies in a CMP ν.[2] Together with $\mathcal{T}$, a policy π induces a probability distribution $\zeta_\nu(\cdot|\pi)$ on trajectories, such that $\zeta_\nu(z|\pi)$ gives the probability that the actor will experience trajectory z if they enact policy π in CMP ν. The expected utility achieved by enacting π in MDP μ is $W_\mu^\pi \triangleq \mathbb{E}_{z\sim\zeta_\mu(\cdot|\pi)}[U(z)]$, and the maximum expected utility that a policy can achieve in μ is $W_\mu^* \triangleq \sup_{\pi\in\mathcal{P}_\mu} W_\mu^\pi$. Sometimes it is useful to consider the expected utility of a trajectory that begins with an arbitrary state and action. $Q_\mu^\pi(s, a)$ gives the expected value of taking action a from state s, and thereafter choosing actions according to π, and $Q_\mu^*(s, a) \triangleq \sup_{\pi\in\mathcal{P}_\mu} Q_\mu^\pi(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}$. Finally, a policy π is said to be *optimal* if $W_\mu^\pi = W_\mu^*$. Any finite MDP is guaranteed to have at least one optimal policy [6]. We use π_μ^* to denote one such policy. Abusing notation, we will sometimes use a subscript θ rather than a subscript μ on W, Q, and π^*, when all aspects of the MDP except the parameters θ that define the reward function are clear from context.

The **inverse reinforcement learning (IRL) problem** can be stated as follows: Given a CMP ν, a trajectory z that an actor has experienced in ν while enacting a fixed policy π, and the actor's discount factor γ, achieve one of the three objectives below:

- **Reward learning**. Infer the actor's reward parameters θ. The solution takes the form of an approximation, $\hat{\theta}$, and is evaluated according to its Euclidean distance from θ: $J_{\text{reward}}(\hat{\theta}; \theta) \triangleq ||\hat{\theta} - \theta||_2$.[3]
- **Apprenticeship learning** [1]. Produce a policy $\hat{\pi}$ that performs as well as possible in ν according to the actor's unknown utility function.[4] The produced policy is evaluated according to the regret of enacting it, compared to enacting an optimal policy: $J_{\text{apprenticeship}}(\hat{\pi}; \theta) \triangleq W_\theta^* - W_\theta^{\hat{\pi}}$.
- **Imitation learning**. Produce a policy $\hat{\pi}$ that replicates the actor's policy π, either for the purpose of imitating the actor or predicting their future behavior. The inferred policy $\hat{\pi}$ can be evaluated at a particular state s according to the following family of objective functions: $J_{\text{imitation}}(\hat{\pi}; \pi, s) \triangleq \mathbb{E}_{a\sim\pi(\cdot|s)}[-\sigma(\hat{\pi}(\cdot|s), a)]$, where σ is a strictly proper scoring rule [5]. In our experiments, we use a normalized version of the logarithmic scoring rule, $\sigma(\hat{\pi}(\cdot|s), a) = \log \frac{\hat{\pi}(a|s)}{\pi(a|s)}$, for which $J_{\text{imitation}}$ reduces to the Kullback-Leibler divergence of $\hat{\pi}(\cdot|s)$ from $\pi(\cdot|s)$.

We focus on Bayesian IRL [2, 7, 8], where the goal is to estimate a joint posterior $\phi(\cdot, \cdot|z)$ on $\Theta \times \mathcal{P}_\nu$ and then choose $\hat{\theta}$ or $\hat{\pi}$ to minimize one of the above objectives in expectation with respect to $\phi(\cdot, \cdot|z)$. In Bayesian IRL, one starts with a joint prior $\phi(\cdot, \cdot)$ on $\Theta \times \mathcal{P}_\nu$, such that $\phi(R, P)$ gives the probability that the actor's reward parameters are in $R \subset \Theta$ and policy is in $P \subset \mathcal{P}_\nu$. The prior can be separated into a prior $\xi(\cdot)$ on Θ, and a collection of conditional distributions $\{\psi(\cdot|\theta)\ \forall\theta \in \Theta\}$ on $\mathcal{P}_\nu$, such that $\phi(R, P) \triangleq \int_R \psi(P|\theta) d\xi(\theta), \forall R \subset \Theta, P \subset \mathcal{P}_\nu$. The posterior can be analogously divided into $\xi(\cdot|z)$ and $\{\psi(\cdot|\theta, z)\ \forall\theta \in \Theta\}$. The statistical model of Bayesian IRL is $\theta \sim \xi(\cdot),\ \pi \sim \psi(\cdot|\theta),\ z \sim \zeta_\nu(\cdot|\pi)$.

3 IRLLA

The problem of inverse reinforcement learning from a learning agent (IRLLA) is a generalization of IRL in which the actor's policy may change over time as a function of their experiences: i.e., as they learn. The actor's policy is updated by a function that we call their **behavior rule**.

Definition 1. *A **behavior rule** is a function β_ν, such that $\beta_\nu(\theta, \gamma, z)$ gives the policy that the actor would enact in CMP ν after experiencing z if their reward parameters were θ and their discount factor were γ.*

A familiar kind of behavior rule maintains a Q-function, which is an estimate of Q_μ^*, and uses it to guide policy selection. We call this kind of behavior rule a **value-based behavior rule** (VBBR). A VBBR has three components: (1) an **initial Q-function**; (2) a **learning rule**, such as Q-learning or SARSA [9], that specifies how to update the Q-function in light

[2] $\mathcal{P}_\nu$ is one of several notations to come that has a subscript CMP. When it is more convenient, we will instead subscript by an MDP (e.g., $\mathcal{P}_\mu$) to refer to the CMP that comprises the MDP.

[3] Alternatively, $\hat{\theta}$ could be evaluated using the Eucidean distance between the induced and the actual reward functions.

[4] This objective is traditionally interpreted as learning how to perform well in μ from the actor, but it can also be interpreted as learning how to properly advise the actor to perform well in μ, which is especially relevant when the actor is learning.

of new experiences; and (3) a **decision rule**, such as softmax, that specifies how to choose a policy based on the Q-function. Softmax specifies a policy π from a Q-function Q as $\pi(a|s) \propto \exp[\eta Q(s,a)]$, with non-negative parameter η controlling how close the produced policy is to being optimal according to Q. Another important kind of behavior rule is the **softmax-optimal behavior rule**, which uses softmax to select a policy based on the optimal Q-function, Q^*_μ.

The statistical model of IRLLA is similar to that of IRL, with one small modification. In IRLLA, "the actor's policy" is imprecise, since it changes over time. On the other hand, the actor's behavior rule is not only well-defined, but also sufficient for the actor's policy at any timestep, when paired with the actor's reward parameters and the actor's trajectory up to that timestep. To see this in our notation, let π_t denote the policy from which a_t was sampled. Then $\pi_t = \beta_\nu(\theta, \gamma, z_t)$. We thus infer a joint posterior over the actor's reward parameters and behavior rule. The statistical model of IRLLA is $\theta \sim \xi(\cdot),\ \beta_\nu \sim \psi(\cdot),\ \pi_t = \beta_\nu(\theta, \gamma, z_t),\ z_{t+1} \sim \zeta_\nu(\cdot|z_t, \pi_t)$, where $\zeta_\nu(z_{t+1}|z_t, \pi_t) = \pi_t(a_t|s_t)\tau(s_{t+1}|a_t, s_t)$ denotes the probability that the $t+1$-step subtrajectory of the actor's trajectory would have been z_{t+1} if they had sampled a_t from $\pi_t(\cdot|s_t)$, given that their t-step subtrajectory is z_t.

Now, suppose that we have (an approximation of) $\phi(\cdot, \cdot|z)$ and can marginalize over behavior rules to obtain $\xi(\cdot|z)$. How can we optimize the reward, apprenticeship, and imitation learning objective functions with respect to $\phi(\cdot, \cdot|z)$? Reward learning is solved with the mean of $\xi(\cdot|z)$, $\hat{\theta} = \mathbb{E}_{\theta\sim\xi(\cdot|z)}[\theta]$, and apprenticeship learning is solved with a policy that is optimal according to the expected reward function, $\mathbb{E}_{\theta\sim\xi(\cdot|z)}[\rho_\theta]$ [7].[5]

Imitation learning is less straightforward. Because the actor's policy changes over time in IRLLA, there are several reasonable ways to generalize the imitation learning objective from traditional IRL to IRLLA. One is to infer the actor's current policy, π_t; another is to predict the actor's next action, a_t; yet another is to predict a sequence of the actor's future policies, $(\pi_t, \pi_{t+1}, ...)$, which may be of interest when interacting with the actor in, for example, a Markov game. In all three cases, inferring π_t is either sufficient or necessary (or both), so we focus on inferring π_t. For simplicity, we write $\pi \sim \phi(\cdot|z)$ to represent generating π by sampling $\theta, \beta \sim \phi(\cdot, \cdot|z)$ and then creating $\pi = \beta(\theta, \gamma, z_t)$ from the sample.

Theorem 1. *Let σ be a strictly proper scoring rule. Then the expected value with respect to $\phi(\cdot, \cdot|z)$ of the imitation objective defined by σ, $\mathbb{E}_{\pi\sim\phi(\cdot|z)}[\mathbb{E}_{a\sim\pi(\cdot|s)}\left[-\sigma(\hat{\pi}(\cdot|s), a)\right]]$, is minimized for all states $s \in \mathcal{S}$ by only $\hat{\pi} = \mathbb{E}_{\pi\sim\phi(\cdot|z)}[\pi]$.*

Thus, regardless of our objective, we wish to compute $\mathbb{E}_{\theta,\beta\sim\phi(\cdot,\cdot|z)}\left[g(\theta, \beta)\right]$ for some function g, where $g(\theta, \beta) \triangleq \theta$ for reward learning, $g(\theta, \beta) \triangleq \rho_\theta$ for apprenticeship learning, and $g(\theta, \beta) \triangleq \beta(\theta, \gamma, z)$ for imitation learning. By self-normalized importance sampling from the prior, we obtain the following consistent estimator [3]:

$$\mathop{\mathbb{E}}_{\theta,\beta\sim\phi(\cdot,\cdot|z_T)}\left[g(\theta, \beta)\right] \approx \sum_{i=1}^{N} g(\theta^i, \beta^i) P_T^i \qquad P_T^i \triangleq \frac{\zeta_\nu(z_T|\theta^i, \beta^i)}{\sum_{j=1}^{N} \zeta_\nu(z_T|\theta^j, \beta^j)} \qquad \left(\theta^i, \beta^i\right) \sim \phi(\cdot, \cdot) \quad \forall i \in \{1, 2, ..., N\} \tag{1}$$

To use this, we must compute $\zeta_\nu(z_T|\theta, \beta)$, the likelihood of (θ, β), which is the probability that in T timesteps the actor experiences z_T in ν if their reward parameters and behavior rule are θ and β, respectively. One can show by rolling out z_T one timestep at a time that $\zeta_\nu(z_T|\theta, \beta) = \left[\tau(s_0) \prod_{t=0}^{T-1} \tau(s_{t+1}|s_t, a_t)\right] \prod_{t=0}^{T-1} \beta(\theta, \gamma, z_t)(a_t|s_t)$. When we compute P_T^i for sample i, the transitions factor in the brackets on the left will appear in the numerator as well as every addend in the denominator, since it doesn't depend on θ or β, so it will cancel out.[6] Thus, $P_T^i = \mathbb{M}(\prod_{t=0}^{T-1} \beta^i(\theta^i, \gamma, z_t)(a_t|s_t))$, where we use $\mathbb{M}(h(i))$ to abbreviate normalization over samples: $\mathbb{M}(h(i)) \triangleq \frac{h(i)}{\sum_{j=1}^{N} h(j)}$. Notice from this that for any time T and for every sample i, $P_{T+1}^i = \mathbb{M}(\beta^i(\theta^i, \gamma, z_T)(a_T|s_T) P_T^i)$. This indicates that P can easily be updated incrementally over time, which enables a simple anytime algorithm, consisting of three phases:

1. **Initialization phase**: Sample $\theta^i, \beta^i \sim \phi(\cdot, \cdot)$ for $i \in \{1, 2, ..., N\}$. For each sample i, compute $\pi_0^i = \beta^i(\theta^i, \gamma, z_0)$.
2. **Update phase**: Upon observing (s_T, a_T), compute $P_{(T+1)}^i = \mathbb{M}(\beta^i(\theta^i, \gamma, z_T)(a_T|s_T) P_T^i)$. $P_0^i = \mathbb{M}(\tau(s_0)) = \frac{1}{N}$.
3. **Query phase**: Upon request, return our approximation of $\mathbb{E}_{\theta,\beta\sim\phi(\cdot,\cdot|z_T)}\left[g(\theta, \beta)\right]$, which is $\sum_{i=1}^{N} g(\theta^i, \beta^i) P_T^i$.

In the special case that we are *a priori* certain that the actor's behavior rule is softmax-optimal, as we might be in the traditional IRL setting, this algorithm reduces to the importance sampling algorithm of Dimitrakakis and Rothkopf [2], which is similar to their Metropolis-Hastings algorithm [8], but with the improvement that all of the expensive planning is done in the initialization phase.

4 Experimental Results

We next demonstrate that our algorithm outperforms its traditional IRL counterpart and, preliminarily, that it is robust to *a priori* uncertainty over the actor's behavior rule. We evaluate our approach in experiments that take place in a

[5] Ramachandran and Amir [7] use a slightly different objective function for apprenticeship learning. The difference doesn't matter.

[6] The transitions term would be 0 only if z_T contains an impossible transition. Since z_T is observed, this will never happen.

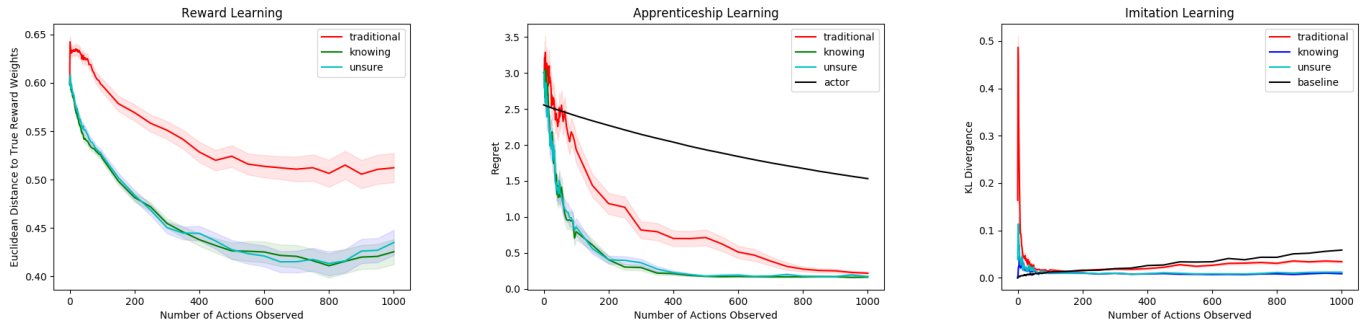


Figure 2: The performance of each instance of the algorithm, averaged over 200 trials. Shading shows one standard error on each side.

Navigation CMP [4], pictured in Figure 1. Each cell in the environment is a state. The blue cell is a terminal state. The actor's reward for a state is determined by the color of that state: 0.4 for blue, 0 for white, -0.4 for red, -0.195 for yellow, and 0.005 for green. The actor's discount factor is 0.95. Their behavior rule is a VBBR, so it consists of an initial Q-function, a learning rule, and a decision rule. They start off completely naive about the CMP's dynamics, thinking that after each action they take, their next state is sampled uniformly at random from $\mathcal{S}$, and their initial Q-function is the result of planning from these beliefs. Their learning rule is Q-learning, with learning rate 0.05. Their decision rule is softmax with parameter $\eta = 20$.

We compare three instances of our algorithm, which differ only in their priors: **traditional**, **knowing**, and **unsure**. Each approximates the posterior with 2,000 samples. They share the same prior over reward functions but have different priors over behavior rules. The prior over reward functions is uniform over the set $\Theta \triangleq \{\theta \in \mathbb{R}^5 : ||\theta||_1 = 1\}$.[7] In addition, all instances are *a priori* certain that the actor's decision rule is softmax, though they are unsure of the parameter η, over which they share an exponential prior with mean 50.

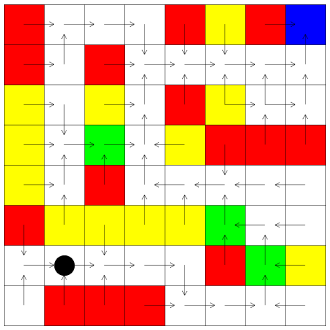


Figure 1: Navigation CMP. The arrows show the actor's optimal policy.

Traditional represents traditional IRL, which is the control in the experiment. It is certain that the actor uses a softmax-optimal behavior rule. **Unsure** starts off 50% sure that the actor uses a softmax-optimal behavior rule, 25% sure that the actor learns via SARSA with learning rate 0.05, and 25% sure that the actor learns via Q-learning with learning rate 0.05.

The three instances are evaluated according to each of the three objectives. In the apprenticeship learning graph, the "actor" bar shows the evaluation of the policy that the actor is using. Being under this bar indicates that we are giving the actor good advice. In the imitation learning graph, the cohorts are evaluated according to the Kullback-Leibler divergence imitation objective at the current state. The "baseline" bar shows the performance of always predicting that the actor chooses actions uniformly at random.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Christos Dimitrakakis and Constantin A Rothkopf. Bayesian multitask inverse reinforcement learning. In *European Workshop on Reinforcement Learning*, pages 273–284. Springer, 2011.

[3] Siem Jan Koopman, Neil Shephard, and Drew Creal. Testing the assumptions behind importance sampling. *Journal of Econometrics*, 149(1):2–11, 2009.

[4] James MacGlashan and Michael L Littman. Between imitation and intention learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[5] Edgar C Merkle and Mark Steyvers. Choosing a strictly proper scoring rule. *Decision Analysis*, 10(4):292–304, 2013.

[6] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[7] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[8] Constantin A Rothkopf and Christos Dimitrakakis. Preference elicitation and inverse reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 34–48. Springer, 2011.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[7] This follows the methodology used by Dimitrakakis and Rothkopf [2]. The softmax parameter η plays the same role as $||\theta||_1$, since they both scale θ. Forcing $||\theta||_1 = 1$ cleans this up by giving η unique power in this regard.

Sparse Imitation Learning for Text Based Games with Combinatorial Action Spaces

Chen Tessler*
Technion Institute of Technology
Haifa, Israel

Tom Zahavy
Technion Institute of Technology
Haifa, Israel

Deborah Cohen
Google Research
Tel-Aviv, Israel

Daniel Mankowitz
Google DeepMind
London, England

Shie Mannor
Technion Institute of Technology
Haifa, Israel

Abstract

We propose a computationally efficient algorithm that combines compressed sensing with imitation learning to solve sequential decision making text-based games with combinatorial action spaces. To do so, we derive a variation of the compressed sensing algorithm Orthogonal Matching Pursuit (OMP), that we call IK-OMP, and show that it can recover a bag-of-words from a sum of the individual word embeddings, even in the presence of noise. We incorporate IK-OMP into a supervised imitation learning setting and show that this algorithm, called Sparse Imitation Learning (Sparse-IL), solves the entire text-based game of Zork1 with an action space of approximately 10 million actions using imperfect, noisy demonstrations.

Keywords: Imitation Learning, Compressed Sensing, Combinatorial Action Spaces

*Corresponding author: chen.tessler@campus.technion.ac.il

1 Introduction

The curses of dimensionality, e.g., large *state* and *action* spaces, pose a major challenge when attempting to apply reinforcement learning (RL) in practice. The goal is thus to find efficient approaches which enable the agent to cope in such scenarios.

Visual based tasks are a common example of infinitely large state spaces for which there exist efficient approaches [Mnih et al., 2015]. Efficiency is achieved through imitation of the human visual system. A network of convolutional filters extracts spatial information from the image and returns a low dimensional dense representation. This efficient representation enables agents to generalize across states and thus efficiently solve complex problems. We turn to the Action Assembly Theory (AAT) [Greene, 2008] in order to understand how to produce an efficient approach for large action spaces.

According to Greene [2008], behavior is described by two essential processes: *representation* and *processing*. Representation refers to the way information is coded and stored in the mind, whereas processing refers to the mental operations performed to retrieve this information. Having good representations of information and an efficient processing procedure allows us to quickly exploit highly rewarding nuances of an environment upon first discovery.

Inspired by the AAT [Greene, 2008], we propose *Sparse Imitation Learning (Sparse-IL)*, a computationally efficient algorithm (see Figure 1). Our approach can be seen as an Encoder-Decoder scheme, in which the state is encoded into a low dimensional continuous embedding vector which is then decoded, resulting in a viable action. This scheme dramatically reduces the time complexity required to select the action.

Our approach Sparse-IL is presented in Figure 1. The **Encoder** is trained using imitation learning. At each state $\mathbf{s}$, the goal of the Encoder is to predict a continuous action $\mathbf{a}_{\text{SoE}}$, which is a dense representation of the action $\mathbf{a}_{\text{env}}$. Specifically, $\mathbf{a}_{\text{SoE}}$ is the sum of the word embeddings of the words in the action/sentence $\mathbf{a}_{\text{env}}$, an efficient representation from which the words can be recovered. This dense representation is then fed into a **Decoder (Retrieval Mechanism)**. Our decoder is composed of 3 major components: (1) Given the vector $\mathbf{a}_{\text{SoE}}$, a Compressed Sensing (CS) algorithm reconstructs the K best Bag-of-Words (BoW - an unordered list of words) actions $\mathbf{a}_{\text{BoW}}$, composed of up to $l = 4$ words. We do this using an algorithm that we term Integer K-Orthogonal Matching Pursuit (IK-OMP). (2) The optimal BoW vector $\mathbf{a}_{\text{BoW}}$, between the suggested K candidates, is selected based on a fitness function. In this work, we consider the reconstruction loss, i.e., the candidate which has the closest embedding to $\mathbf{a}_{\text{SoE}}$. Finally, the BoW vector $\mathbf{a}_{\text{BoW}}$ is fed into (3) a language model to yield an action sentence $\mathbf{a}_{\text{env}}$ which is a valid environment action.

Main contributions: We propose a computationally efficient algorithm called Sparse-IL that combines CS with imitation learning to solve natural language tasks with combinatorial action spaces. We demonstrate that Sparse-IL can solve the entire game of Zork1, for the first time, when considering a combinatorial action space of approximately 10 million actions, using noisy, imperfect demonstrations.

2 Background & Problem Setting

Our work focuses on the domain of **Zork1**, an interactive text-based game [Côté et al., 2018], for which an example is provided in Figure 2. We model text-based games using the standard RL formulation [Sutton and Barto, 1998], where the states are paragraphs and the actions are sentences. We consider two tasks (1) the 'Troll Quest', a sub-task within Zork1 in which the agent must collect a sword, find a troll and kill it with the sword, and (2) 'Open Zork' in which the goal is to solve the entire game of Zork1, an open problem until now. In Zork1, the agent receives reward for advancing the story-line, this includes reaching certain checkpoints and collecting various items. An optimal policy will receive a reward of 45 in the 'Troll Quest' and 350 in 'Open Zork'.

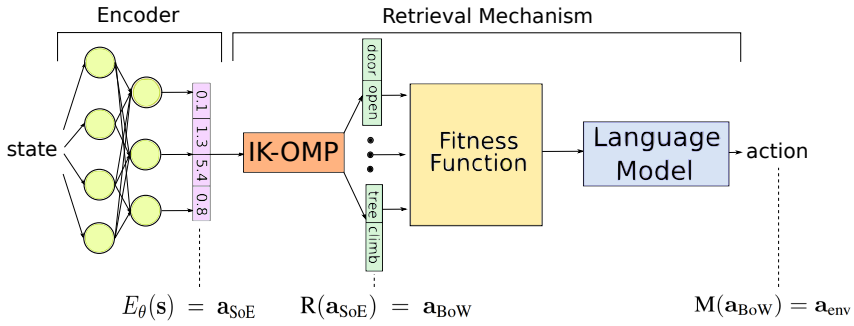


Figure 1: A graphical representation of our algorithm.

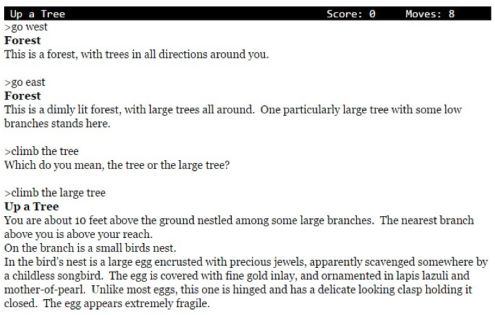


Figure 2: Zork1 example screen.

While Zork1 has many challenges, we focus in this work on the **large and complex action space**. As the actions are free-text, it is possible to issue complex commands such as "drop all except the egg and sword" and "look under the rug". The walkthrough (optimal policy) we used, solves the game in ≈ 400 steps, using $d = 112$ unique words and sentences composed of up to $l = 4$ words. Thus, the number of possible, unordered, word combinations are $|A| = d^l/l!$, i.e., the dictionary size to the power of the maximal sentence length, divided by the number of possible permutations. This results in approximately 10 million possible actions.

In order to isolate the problem, we solve the task using imitation learning. Similar to Silver et al. [2016], by removing the element of self-play, we are able to deepen our analysis and look into the the various components of our approach. We show that our approach is capable of solving the entire game of Zork1 when considering the entire set of possible actions. This is performed using an efficient Encoder-Decoder scheme, in which the encoding is the sum of word embeddings $\mathbf{a}_{\text{SoE}}$, a continuous dense vector. In this work, we use pre-trained word embeddings. The decoding is performed using a CS approach (IK-OMP, Section 3) to recover the sparse BoW representation of the action $\mathbf{a}_{\text{BoW}}$ from the sum of word embeddings $\mathbf{a}_{\text{SoE}}$. Note that this algorithmic approach does not require training and is not domain specific.

CS is concerned with recovering sparse representations, such as $\mathbf{a}_{\text{BoW}}$ from low-dimensional, dense, continuous representations, here, $\mathbf{a}_{\text{SoE}}$. The former can be written as a linear combination of the latter:

$$\min ||\mathbf{a}_{\text{BoW}}||_0 \text{ subject to } \mathbf{D}\mathbf{a}_{\text{BoW}} = \mathbf{a}_{\text{SoE}} \ , \tag{1}$$

where $\mathbf{D}$ is the embeddings matrix, whose i-th column contains the embeddings vector of the i-th token in the dictionary D. We refer the reader to Elad [2010] for an overview which covers recovery guarantees and ability to cope with noise, in addition to an overview of the existing approaches.

A common recovery algorithm for solving (1) is OMP [Blumensath and Davies, 2008]. The popular OMP algorithm proceeds by iteratively finding the dictionary entry with the highest correlation to the signal residual, computed by subtracting the contribution of a partial estimate of $\mathbf{a}_{\text{BoW}}$ from $\mathbf{a}_{\text{SoE}}$. The coefficients over the selected support set are then chosen so as to minimize the residual error. In this work, we derive our own variant OMP, referred to IK-OMP and presented in Section 3.

3 Method

Our method combines CS with imitation learning in order to produce an efficient approach to coping with combinatorial action space. The Encoder $\text{E}(\mathbf{s}) = \mathbf{a}_{\text{SoE}}$, a Convolutional Neural Network (CNN) that is suited to NLP tasks [Kim, 2014], is trained to predict the sum of word embeddings of the demonstrated action $\mathbf{a}_{\text{env}}$, e.g., $\mathbf{a}_{\text{SoE}} = \sum_{w \in \mathbf{a}_{\text{env}}} \text{embedding}(w)$. A **CS (Decoder)** algorithm $\text{R}(\mathbf{a}_{\text{SoE}}) = \mathbf{a}_{\text{BoW}}$, namely IK-OMP (see explanation below), reconstructs the most plausible BoW vector $\mathbf{a}_{\text{BoW}}$, e.g., it recovers the words which compose $\mathbf{a}_{\text{SoE}}$. Finally, the BoW vector $\mathbf{a}_{\text{BoW}}$ is fed into a **Language Model** $\text{M}(\mathbf{a}_{\text{BoW}}) = \mathbf{a}_{\text{env}}$. The goal of the language model is to output the most likely ordering of the provided words, such that it yields a grammatically correct sentence. In this paper, we use a rule based approach. Our rules are relatively simple, yet work surprisingly well - e.g., given a verb and an object, the verb comes before the object - e.g., ['sword', 'take'] $\mapsto$ 'take sword'.

The standard approaches compute the Q value for each action [Mnih et al., 2015, He et al., 2016] or compare the predicted embedding to all the possible action embeddings [Dulac-Arnold et al., 2015], hence their time complexity $\mathcal{O}(|A|) = \mathcal{O}(d^l/l!)$ operations at each step. In our approach, the Encoder is a feed-forward neural network which is invariant to the size of the action space $\mathcal{O}(1)$, and the Decoder (IK-OMP) enjoys a complexity of $\mathcal{O}(l * k * d)$, as it is required to reconstruct an l length sentence using a given dictionary $\mathbf{D}$.

Integer K-OMP (IK-OMP) is an alternative CS algorithm, which we introduce in this work. We observe that the standard CS algorithms attempt to recover a continuous BoW vector. However, as the BoW represents a sentence, it must reside in $\in \mathbb{N}^d$. IK-OMP leverages this prior knowledge by adding an integer constraint. Additionally, as OMP iteratively adds atoms to the recovered support, the choice of a new element in an iteration is blind to its effect on future iterations. To mitigate this phenomenon, we suggest to use a beam-search procedure which maintains a set of K possible candidates at each iteration. Our approach is presented in Algorithm 1. Given the final K candidates, the selected BoW vector is the one which minimizes

$$\underset{\mathbf{a}^i_{\text{BoW}} \in [\mathbf{a}^1_{\text{BoW}}, \ldots, \mathbf{a}^k_{\text{BoW}}]}{\arg\min} ||\mathbf{a}_{\text{SoE}} - \mathbf{D}\mathbf{a}^i_{\text{BoW}}||_2^2 \ .$$

4 Experiments

We focus on two experiments, in which we make use of a walkthrough of the game - a sequence of 400 actions required to solve the game. Our *first experiment* evaluates the Decoder under a controlled setting, specifically, we test the ability of

Algorithm 1 IK-OMP

Input: Measurement vector $\mathbf{y} \in \mathbb{R}^m$, dictionary $\mathbf{D} \in \mathbb{R}^{m \times d}$, maximal number of characters L and beam width K
Initial solutions $\mathbf{X}^0 = [\mathbf{0}_d, \ldots, \mathbf{0}_d]$
for $l = 1, L$ **do**
  for $i \in [1, \ldots, k]$ **do**
    Extend: Append $\mathbf{X}_i^{l-1} + \mathbf{1}_j, \forall j \in [1, ..., d]$ to $\mathbf{X}^{l-1}$
  end for
  Trim: $\mathbf{X}^l = \text{K-}\arg\min_{\mathbf{X}_i \in \mathbf{X}^{l-1}} ||\mathbf{y} - \mathbf{D}\mathbf{X}_i||_2^2$
end for
return $\mathbf{X}^L$

Figure 3: **Runtime comparison** of various CS approaches.

Algorithm	Seconds per Action
OMP	0.008
IK-OMP, K=1	0.008
IK-OMP, K=3	0.021
IK-OMP, K=20	0.166
IK-OMP, K=112	1.116
FISTA (Basis Pursuit)	0.881

the various CS approaches to reconstruct the original action given a noisy and imperfect demonstration. In the *second experiment*, we add the Encoder, which is trained using imitation learning, and test the performance of the system as a whole (Encoder + Decoder).

4.1 Compressed Sensing

We begin by comparing several CS approaches. At each state $\mathbf{s}$, we take the ground-truth action $\mathbf{a}_{\text{env}}(\mathbf{s})$ from the walkthrough, calculate the sum of word embeddings $\mathbf{a}_{\text{SoE}}(\mathbf{s})$, add noise ϵ and test the ability of various CS methods to reconstruct $\mathbf{a}_{\text{env}}(\mathbf{s})$. For each method, we compare the *run-time* (Figure 3), the *reconstruction accuracy* and the *reward gained* in the presence of noise (Figure 4). Specifically, the measured action is $\mathbf{a}_{\text{mes}}(\mathbf{s}) = \mathbf{a}_{\text{SoE}}(\mathbf{s}) + \epsilon$, where $\epsilon \sim N(0, 1)$ is normalized based on $\mathbf{a}_{\text{SoE}}(\mathbf{s})$ and the signal to noise ratio (SnR). While *accuracy* and *reward* might seem similar, an inaccurate reconstruction at an early stage results in an immediate failure, even when the accuracy seems high.

We compare 4 CS methods: FISTA (Basis Pursuit, Arora et al. [2018]), OMP and IK-OMP. The dictionary is composed of $d = 112$ possible words which can be used in the game. The dimension of the embedding is $m = 50$ and the sentence length is limited to at most 4 words. This yields a total number of ≈ 10 million actions, from which the agent must choose.

Our experiments show that our suggested approach, IK-OMP, outperforms the various baselines (OMP and FISTA) both in terms of run-time and performance and is capable of correctly reconstructing the original action $\mathbf{a}_{\text{BoW}}$, even in the presence of relatively large noise. This gives evidence that the integer prior, in particular, and the beam search strategy significantly improve the sparse recovery performance when considering textual signals.

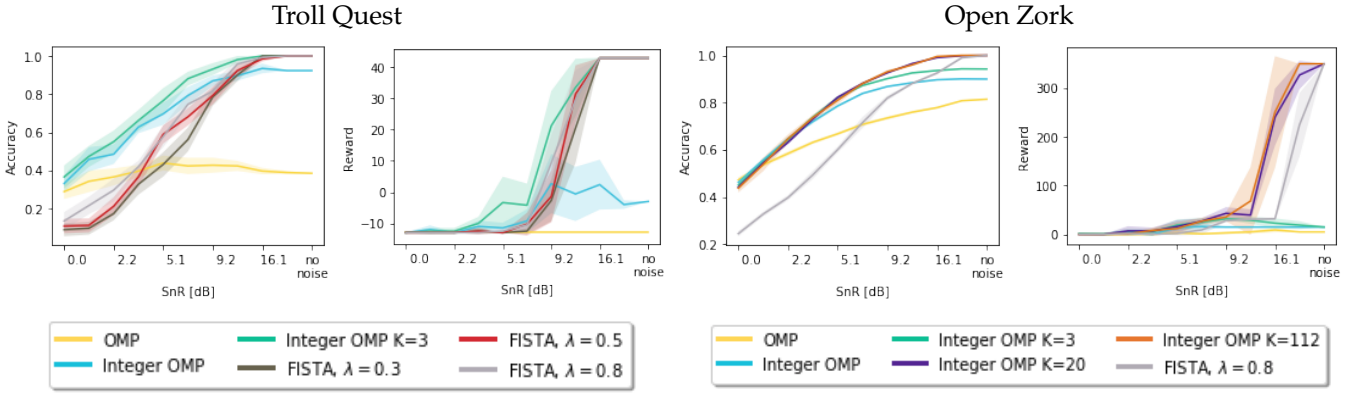


Figure 4: **Compressed Sensing**: Comparison of the accuracy, and accumulated reward, of the various reconstruction algorithms on the 'Troll Quest' and in 'Open Zork'. FISTA with $\lambda = 0.8$ is selected based on the comparison in the 'Troll Quest'. The SnR denotes the ratio between the norm of the original signal $\mathbf{a}_{\text{SoE}}$ and that of the added noise.

4.2 Imitation Learning

Given a data set of state-action pairs $(\mathbf{s}, \mathbf{a}_{\text{env}})$, provided by an expert, the goal is to learn a policy that achieves the best performance possible. In an imitation learning (IL) setting, the goal is to learn to predict the action $\mathbf{a}_{\text{env}}$ the expert took.

The embedding network $\text{E}_\theta(\mathbf{s})$ is trained to predict (imitate) the sum of word embeddings $\mathbf{a}_{\text{SoE}}(\mathbf{s})$ of the demonstrated actions $\mathbf{a}_{\text{env}}(\mathbf{s})$, by minimizing the mean squared error between the predicted actions and those demonstrated. We consider three setups: (1) Perfect demonstrations, where the goal is to test errors due to architecture capacity and function

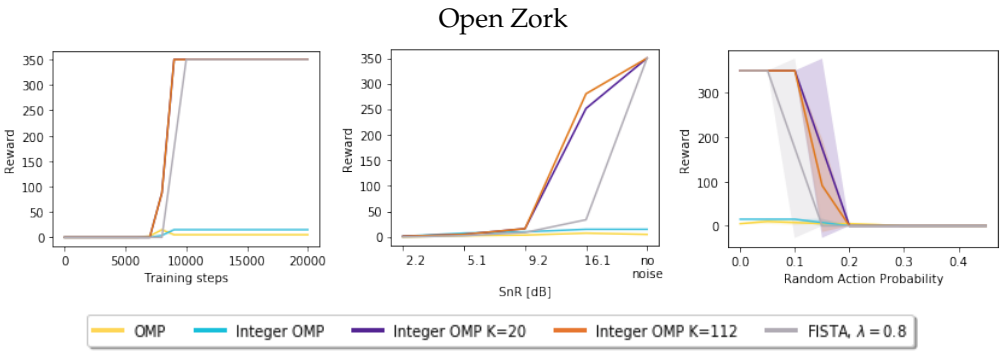


Figure 5: **Sparse Imitation Learning:** Comparison of the accuracy of each reconstruction algorithm on an agent trained using imitation learning to solve the entire game. **Left** - Training, **center** - Gaussian Noise, **right** - Discrete Action Noise. In the left graph, IK-OMP with K=20 and K=112 result in identical performance.

approximation, (2) Gaussian noise, $\mathbf{a}_{\text{mes}}(\mathbf{s}) = \mathbf{a}_{\text{SoE}}(\mathbf{s}) + \epsilon$, where the noise is encountered in the embedding space, and (3) discrete-action noise, in which, with a fixed probability (the x-axis in Figure 5), we replace the ground truth of each sample with a random, different, action.

Our results, depicted in Figure 5, show that the combination of CS and imitation learning is capable of solving the entire game of Zork1 (the maximal reward is 350), even in the presence of discrete-action noise. We observe that IK-OMP outperforms the other methods and shows improved robustness to noise. In the Gaussian noise test, we observe that all methods are incapable of handling imperfect demonstrations - this is attributed to the additive noise, that of the approximation error of the network and the Gaussian noise present during training.

5 Conclusions

In this work, we tackled the problem of combinatorial action spaces. We demonstrated that IK-OMP, an improved variant of the CS algorithm OMP, is able to recover $\mathbf{a}_{\text{BoW}}$ given noisy observations of $\mathbf{a}_{\text{SoE}}$. In addition, we presented a combined approach of CS with imitation learning. Our approach, Sparse Imitation Learning, is capable of solving the entire game of Zork1 *for the first time*, even when provided imperfect demonstrations. In future work, we would like to consider replacing the fitness function, which selects between the K candidate actions, with a critic network, such that our approach will be able to improve the performance over the noisy imitation results.

References

Sanjeev Arora et al. A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and lstms. 2018.

Thomas Blumensath and Mike E Davies. Gradient pursuits. *IEEE Transactions on Signal Processing*, 56(6):2370–2382, 2008.

Marc-Alexandre Côté et al. Textworld: A learning environment for text-based games. *arXiv*, 2018.

Gabriel Dulac-Arnold et al. Deep reinforcement learning in large discrete action spaces. *arXiv*, 2015.

Michael Elad. *Sparse and Redundant Representations: From Theory to Applications*. Springer, 2010.

John O Greene. Action assembly theory. *The International Encyclopedia of Communication*, 2008.

Ji He et al. Deep reinforcement learning with a natural language action space. In *ACL*, 2016.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv*, 2014.

Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

David Silver et al. Mastering the game of go with deep neural networks and tree search. *nature*, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Action Robust Reinforcement Learning and Applications in Continuous Control

Chen Tessler* †
Technion Institute of Technology
Haifa, Israel

Yonathan Efroni*
Technion Institute of Technology
Haifa, Israel

Shie Mannor
Technion Institute of Technology
Haifa, Israel

Abstract

A policy is said to be robust if it maximizes the reward while considering a bad, or even adversarial, model. In this work we formalize a new criterion of robustness to action uncertainty. Specifically, we consider a scenario in which the agent attempts to perform an action $\mathbf{a}$, and with probability α, an alternative adversarial action $\bar{\mathbf{a}}$ is taken. We show that our criterion is related to common forms of uncertainty in robotics domains, such as the occurrence of abrupt forces, and suggest algorithms in the tabular case. Building on the suggested algorithms, we generalize our approach to deep reinforcement learning (DRL) and provide extensive experiments in the various MuJoCo domains. Our experiments show that not only does our approach produce robust policies, but it also improves the performance in the absence of perturbations. This generalization indicates that action-robustness can be thought of as implicit regularization in RL problems.

Keywords: Robustness, Reinforcement Learning, Continuous Control

*equal contribution

†chen.tessler@campus.technion.ac.il

1 Introduction

Recent advances in Reinforcement Learning (RL) have demonstrated its potential in real-world deployment. However, since in RL it is normally assumed that the train and test domains are identical, it is not clear how a learned policy would generalize under small perturbations. For example, consider the task of robotic manipulation in which the task is to navigate towards a goal. As the policy is trained on a specific parameter set (mass, friction, etc...), it is not clear what would happen when these parameters change, e.g., if the robot is slightly lighter/heavier.

The advantage of robust policies is highlighted when considering imperfect models, a common scenario in real world tasks such as autonomous vehicles. Even if the model is trained in the real world, certain variables such as traction, tire pressure, humidity, vehicle mass and road conditions may vary over time. These changes affect the dynamics of our model, a property which should be considered during the optimization process. Robust MDPs (Nilim & El Ghaoui, 2005; Iyengar, 2005) tackle this issue by solving a max-min optimization problem over a set of possible model parameters, an uncertainty set, e.g., the range of values which the vehicle's mass may take - the goal is thus to maximize the reward, with respect to (w.r.t.) the worst possible outcome.

Previously, Robust MDPs have been analyzed extensively in the theoretical community, in the tabular case (Nilim & El Ghaoui, 2005; Iyengar, 2005) and under linear function approximation (Tamar et al., 2013). However, as these works analyze uncertainty in the transition probabilities: (i) it is not clear how to obtain these uncertainty sets, and (ii) it is not clear if and how these approaches may be extended to non-linear function approximation schemes, e.g., neural networks. Recently, this problem has been tackled, empirically, by the Deep RL community (Pinto et al., 2017). While these approaches seem to work well in practice, they require access and control of a simulator and are not backed by theoretical guarantees - a well known problem in adversarial training (Barnett, 2018).

Our approach tackles these problems by introducing a natural way to define robustness - robustness w.r.t. action perturbations - a scenario in which the agent attempts to perform an action and due to disturbances, such as noise or model uncertainty, acts differently than expected. In this work, we consider a probabilistic robustness criterion: given an action provided by the policy the *Probabilistic Action Robust MDP (PR-MDP, Section 3)* criterion considers the case in which, with probability α, a different possibly adversarial action is taken. This criterion is strongly correlated to real world uncertainty, for instance, abrupt interruptions such as a sudden push.

2 Preliminaries

2.1 Markov Decision Process

We consider the framework of infinite-horizon discounted Markov Decision Process (MDP) with continuous action space. An MDP is defined as the 5-tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ Puterman (1994), where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a compact and convex action metric space. We assume $P \equiv P(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ is a transition kernel and is weakly continuous in $\mathbf{a}$, $R \equiv r(\mathbf{s}, \mathbf{a})$ is a reward function continuous in a, and $\gamma \in (0, 1)$. Let $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ be a stationary policy, where $\mathcal{P}(\mathcal{A})$ is the set of probability measures on the Borel sets of $\mathcal{A}$. We denote Π as the set of stationary deterministic policies on $\mathcal{A}$, i.e., if $\pi \in \Pi$ then $\pi : \mathcal{S} \to \mathcal{A}$, and $\mathcal{P}(\Pi)$ as the set of stationary stochastic policies. Let $v^\pi \in \mathbb{R}^{|\mathcal{S}|}$ be the value of a policy π, defined in state $\mathbf{s}$ as $v^\pi(\mathbf{s}) \equiv \mathbb{E}^\pi[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s}]$, where $\mathbf{a}_t \sim \pi(\mathbf{s}_t)$ is a random-variable, $\mathbb{E}^\pi$ denotes expectation w.r.t. the distribution induced by π and conditioned on the event $\{\mathbf{s}_0 = \mathbf{s}\}$.

3 Probabilistic Action Robust MDP[1]

Definition 1. *Let $\alpha \in [0, 1]$. A Probabilistic Action Robust MDP is defined by the 5-tuple of an MDP (see Section 2.1). Let $\pi, \bar{\pi}$ be policies of an agent an adversary. We define their probabilistic joint policy $\pi^{\text{mix}}_{P,\alpha}(\pi, \bar{\pi})$ as $\forall s \in \mathcal{S},\ \pi^{\text{mix}}_{P,\alpha}(\mathbf{a} \mid \mathbf{s}) \equiv (1 - \alpha)\pi(\mathbf{a} \mid \mathbf{s}) + \alpha\bar{\pi}(\mathbf{a} \mid \mathbf{s})$.*

Let π be an agent policy. As opposed to standard MDPs, the value of the policy is defined by $v^\pi_{P,\alpha} = \min_{\bar{\pi} \in \Pi} \mathbb{E}^{\pi^{\text{mix}}_{P,\alpha}(\pi, \bar{\pi})}[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$, where $\mathbf{a}_t \sim \pi^{\text{mix}}_{P,\alpha}(\pi(\mathbf{s}_t), \bar{\pi}(\mathbf{s}_t))$. The optimal probabilistic robust policy is the optimal policy of the PR-MDP

$$\pi^*_{P,\alpha} \in \arg\max_{\pi \in \mathcal{P}(\Pi)} \min_{\bar{\pi} \in \Pi} \mathbb{E}^{\pi^{\text{mix}}_{P,\alpha}(\pi, \bar{\pi})}[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]. \tag{1}$$

Simply put, an optimal probabilistic robust policy is optimal w.r.t. a scenario in which, with probability α, an adversary takes control and performs the worst possible action. This approach formalizes a possible inability to control the system and to perform the wanted actions.

[1] Proofs are provided in the full version: `https://arxiv.org/abs/1901.09184`

Algorithm 1 Probabilistic Robust PI

Initialize: $\alpha, \bar{\pi}_0, k = 0$
while not changing **do**
  $\pi_k \in \arg\max_{\pi'} v^{\pi^{\text{mix}}_{P,\alpha}(\pi', \bar{\pi}_k)}$
  $\bar{\pi}_{k+1} \in \arg\min_{\bar{\pi}} r^{\bar{\pi}} + \gamma P^{\bar{\pi}} v^{\pi^{\text{mix}}_{P,\alpha}(\pi_k, \bar{\pi}_k)}$
  $k \leftarrow k + 1$
end while
Return π_{k-1}

Algorithm 2 Soft Probabilistic Robust PI

Initialize: $\alpha, \eta, \bar{\pi}_0, k = 0$
while criterion is not satisfied **do**
  $\pi_k \in \arg\max_{\pi'} v^{\pi^{\text{mix}}_{P,\alpha}(\pi', \bar{\pi}_k)}$
  $\bar{\pi} \in \arg\min_{\bar{\pi}'} \left\langle \bar{\pi}', \nabla_{\bar{\pi}} v^{\pi^{\text{mix}}_{P,\alpha}(\pi_k, \bar{\pi})} \mid_{\bar{\pi}=\bar{\pi}_k} \right\rangle$
  $\bar{\pi}_{k+1} = (1 - \eta)\bar{\pi}_k + \eta\bar{\pi}$
  $k \leftarrow k + 1$
end while
Return π_{k-1}

In-order to obtain the optimal probabilistic robust policy, one needs to solve the zero-sum game as defined in (1). It is well known that any zero-sum game has a well defined value on the set of stochastic policies, but not always on the set of deterministic policies. Interestingly, and similarly to regular MDPs, the optimal policy of the PR-MDP is a deterministic one as the following proposition asserts.

Proposition 1. *For PR-MDP, there exists an optimal policy which is stationary and deterministic, and strong duality holds in* Π, $v^*_{P,\alpha} = \max_{\pi\in\Pi} \min_{\bar{\pi}\in\Pi} \mathbb{E}^{\pi^{\text{mix}}_{P,\alpha}(\pi,\bar{\pi})}[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)] = \min_{\bar{\pi}\in\Pi} \max_{\pi\in\Pi} \mathbb{E}^{\pi^{\text{mix}}_{P,\alpha}(\pi,\bar{\pi})}[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$.

3.1 Probabilistic Action Robust and Robust MDPs

Although the approach of PR-MDP might seem orthogonal to the that of Robust MDPs, the former is a specific case of the latter. By using the PR-MDP criterion, a class of models is implicitly defined, and the probabilistic robust policy is optimal w.r.t. the worst possible model in this class. To see the equivalence, define the following class of models,

$$\mathcal{P}_\alpha = \{(1 - \alpha)P + \alpha P^\pi : \mathcal{P}(\Pi)\} \ , \ \mathcal{R}_\alpha = \{(1 - \alpha)r + \alpha r^\pi : \pi \in \mathcal{P}(\Pi)\}.$$

A probabilistic robust policy, which solves (1), is also the solution to $\pi^*_{P,\alpha} \in \arg\max_{\pi'\in\Pi} \min_{P\in P_\alpha, r\in\mathcal{R}_\alpha} \mathbb{E}^{\pi'}_P[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$, where $\mathbb{E}^\pi_P$ is the expectation of policy π when the dynamics are given by P. This relation explicitly shows that $\pi^*_{P,\alpha}$ is also optimal w.r.t. the worst model in the class $\mathcal{P}_\alpha, \mathcal{R}_\alpha$, which is convex and rectangular uncertainty set , and formalizes the fact that PR-MDP is a specific instance of RMDP.

3.2 Policy Iteration Schemes for PR-MDP

The Probabilistic Robust PI (PR-PI, Algorithm 1) is a two player PI scheme adjusted to solving a PR-MDP. PR-PI repeats two stages, (i) given a fixed adversary strategy, it calculates the optimal counter strategy, and (ii) it solves the 1-step greedy policy w.r.t. the value of the agent and adversary mixture policy.

The Soft Probabilistic Robust PI (Soft PR-PI, Algorithm 2) is updated using gradient information, unlike the PR-PI. Instead of updating the adversary policy using a 1-step greedy update, the adversary policy is updated using a Frank-Wolfe update. The Franke-Wolfe update, similar to the gradient-projection method, finds a policy which is within the set of feasible policies. Since a convex mixture of two policies is a valid policy, the new policy is ensured to be a valid one.

While the two algorithms might seem disparate, Soft PR-PI merely generalizes the 'hard' updates of PR-PI to 'soft' ones. This statement is formalized in the following proposition, a direct consequence of Theorem 1 in Scherrer & Geist (2014).

Proposition 2. *Let* $\pi, \bar{\pi}$ *be general policies. Then,* $\arg\min_{\bar{\pi}'\in\Pi} r^{\bar{\pi}'} + \gamma P^{\bar{\pi}'} v^{\pi^{\text{mix}}_{P,\alpha}(\pi,\bar{\pi})} = \arg\min_{\bar{\pi}'\in\Pi} \left\langle \bar{\pi}', \nabla_{\bar{\pi}} v^{\pi^{\text{mix}}_{P,\alpha}(\pi,\tilde{\pi})} \mid_{\tilde{\pi}=\bar{\pi}} \right\rangle$.

Note that when $\eta = 1$ we have that both Algorithms 1 and 2 are identical. In addition, the following result shows that both algorithms converge to the unique optimal value of the Nash-Equilibrium.

Theorem 3. *Denote by* $v_k \stackrel{def}{=} v^{\pi^{\text{mix}}_{P,\alpha}(\pi_k,\bar{\pi}_k)}$. *Then, for any* $\eta \in (0, 1]$, *in Algorithm 2 (or* $\eta = 1$ *for Algorithm 1),* v_k *contracts toward* $v^*_{P,\alpha}$ *with coefficient* $(1 - \eta + \gamma\eta)$, *i.e.,*

$$||v_k - v^*_{P,\alpha}||_\infty \leq (1 - \eta + \gamma\eta)||v_{k-1} - v^*_{P,\alpha}||_\infty \ .$$

4 Experiments

Our approach adapts the Soft PR-PI algorithm to the high dimensional scenario. We train both an actor and an adversary using gradient ascent/descent. As, in practice, it is hard to measure convergence; we train the actor for N gradient steps followed by a single adversary gradient step.

We focus on a robust variant of DDPG which we call Action-Robust DDPG. In DDPG Lillicrap et al. (2015), the output of the policy network is a deterministic policy $\mu_\theta : \mathcal{S} \to \mathcal{A}$. As opposed to DDPG, in Action-Robust DDPG two networks output two deterministic policies, the actor and adversary policies, denoted by μ_θ and $\bar{\mu}_{\bar{\theta}}$. The critic is trained to estimate the q-function of the joint-policy, e.g., $\pi^{\text{mix}}_{P,\alpha}(u \,|\, s; \theta, \bar{\theta}) = (1-\alpha)\delta(u - \mu_\theta(s)) + \alpha\delta(u - \bar{\mu}_{\bar{\theta}}(s))$.

Proposition 4. *Let $\mu_\theta, \bar{\mu}_{\bar{\theta}}$ be the agent's and adversary's deterministic policies, respectively. Let $\pi(\mu_\theta, \bar{\mu}_{\bar{\theta}})$ be the joint policy given the agent and adversary policies. i.e., $\pi = \pi^{\text{mix}}_{P,\alpha}$.*

Let $J(\pi(\mu_\theta, \bar{\mu}_{\bar{\theta}})) = \mathbb{E}_{\mathbf{s}\sim\rho^\pi}[v^\pi(s)]$ be the performance objective. The gradient of the actor and adversary parameters is:

$$\nabla_\theta J(\pi(\mu_\theta, \bar{\mu}_{\bar{\theta}})) = (1-\alpha)\mathbb{E}_{\mathbf{s}\sim\rho^\pi}[\nabla_\theta \mu_\theta(\mathbf{s}) \nabla_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})] \;\; , \;\; \nabla_{\bar{\theta}} J(\pi(\mu_\theta, \bar{\mu}_{\bar{\theta}})) = \alpha\mathbb{E}_{\mathbf{s}\sim\rho^\pi}[\nabla_{\bar{\theta}} \bar{\mu}_{\bar{\theta}}(\mathbf{s}) \nabla_{\mathbf{a}} Q^\pi(\mathbf{s}, \bar{\mathbf{a}})] \;\; .$$

The Action-Robust DDPG has several parameters that need to be tuned. The scalar α which measures the adversary's 'strength', the type of exploration scheme, and the ratio between the number of gradient steps of the actor and adversary. We tune these parameters by a naive parameter scan; we find optimal parameter given the others are fixed, as we show in Figure 1. The parameters we choose are the ones that attain the best performance on the Hopper domain.

We test the selected configuration, across a range of continuous control tasks in MuJoCo, on a robustness to abrupt forces and model uncertainty task (see Figure 2). We end the experiments section with an attempt to provide some additional insight into how and why the approach works.

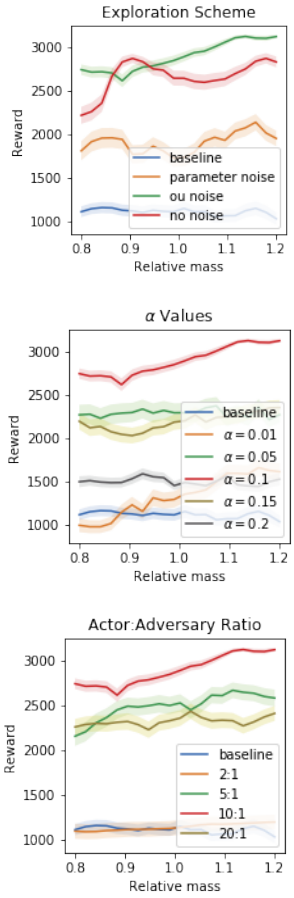


Figure 1: Hopper-v2: Robust parameter comparison.

4.1 Training

We test various α values, actor update steps N and exploration schemes. Each configuration, is trained on 5 random seeds and evaluated across 100 episodes. The evaluation is performed without adversarial perturbations, on a range of mass values not encountered during training. We compare to the non-robust DDPG with parameter space noise, the best performing method in the Hopper-v2 domain.

Figure 1 presents the ablation results. It is clear that the best performing exploration scheme is OU noise, the perturbation parameter α is 0.1 (optimal under the worst case mass perturbation during the evaluation) and the number of Actor gradient steps (per each single Adversary step) is 10. An interesting observation is that even when lacking additional external noise, the agent reaches high performance, this suggests that the adversary induces an efficient exploration scheme.

4.2 Testing on various MuJoCo domains

Figure 2 presents our results, on various MuJoCo domains. It is apparent that while in the Hopper-v2 domain, the PR-MDP outperformed the NR-MDP criterion; this does not hold on all domains. Moreover, in most of the domains, both operators outperform the baseline, both in terms of robustness and in terms of performance in the absence of perturbations.

Failures: It is also important to acknowledge the scenarios in which our algorithm does not outperform the baseline. Such an example is the InvertedPendulum domain, in which the performance of the robust operators was found to be inferior to that of its non-robust counterpart. We find two possible explanations for this phenomenon (i) the parameters are pre-selected based on the performance in the Hopper domain. (ii) Specifically in the InvertedPendulum domain, where the task is to balance a pole, an adversary which is too strong (large α value) prevents the agent from solving the task.

4.3 Diving Deeper

We attempt to analyze the behavior of our criteria (Figure 3) by asking two questions: (i) Does the performance increase due to the added perturbations from the adversary, or does the operator itself induce a prior, e.g., regularization, on the policy which leads to improved performance. (ii) How close is the empirical behavior to its theoretical counterpart.

Off-Policy Action Robustness: In previous experiments, during training, the action was drawn from the joint policy of the agent and adversary, where the joint policy is specified in the PR-MDP approaches (see Definition 1). A natural alternative approach is to *act with the actor's policy*, yet, to acquire an action-robust policy in an off-policy fashion. Meaning, use the same algorithms while obtaining the data without the effect of the adversary. A possible advantage of

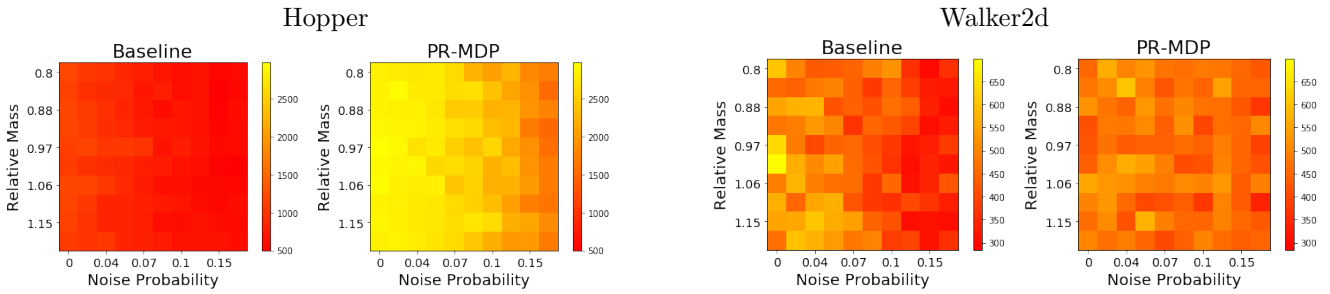


Figure 2: Robustness to model uncertainty.

such an approach is minimizing the number of bad actions (since the adversary does not intervene), while still benefiting from the presence of robust learning.

Figure 3 presents the results of this experiment. Surprisingly, the off-policy nature, e.g., not acting with the adversarial policy, leads to performance degradation.

Does MaxMin equal MinMax? While so far we trained our agent through N actor updates followed by a single adversary gradient update, this corresponds to the MinMax operator, in theory the opposite should result in an identical performance for the PR-MDP approach. Experimentally (Figure 3), we see that indeed there exists a large correlation between the theoretical results and those seen in practice. The results are similar to the regular ratio (Figure 1) with slight improvement.

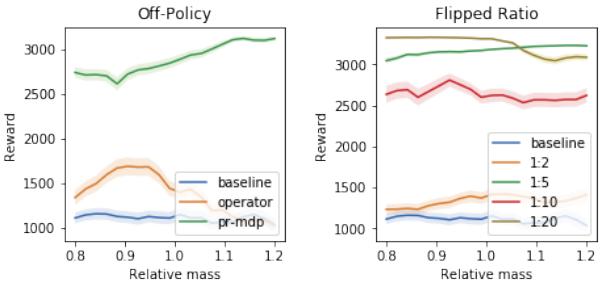


Figure 3: (Left) Testing Off-Policy Action-Robustness, and (Right) Solving the MaxMin operator.

5 Summary

We have presented a new criteria for robustness, the Probabilistic Action Robust MDP. Additionally; we developed the Soft PR-PI (Algorithm 2), a policy iteration scheme for solving PR-MDPs. Building upon the Soft PR-PI algorithm, we presented a deep reinforcement learning approach, which is capable of solving our criterion. We analyzed how the various parameters affect the behavior and how the empirical results correlate with the theoretical approach. Most importantly, we notice that not only does training with our criterion result in robust policies, but our approach improves performance even in the absence of perturbations.

Lastly, for solving an action-robust policy, there is *no need in providing an uncertainty set*. The approach requires only a scalar value, namely α (or possibly a state-dependent $\alpha(s)$), which *implicitly* defines an uncertainty set (see Section 3.1). This is a major advantage compared to standard robust approaches in RL and control, which, to the best of our knowledge, require a distribution over models or perturbations. Of course, this benefit is also a restriction - the Action Robust approach is unable to handle any kind of worst-case perturbations. Yet, due to its simplicity, and its demonstrated performance, it is worthwhile to be considered by an algorithm designer.

References

Barnett, S. A. Convergence problems with generative adversarial networks (gans). *arXiv preprint arXiv:1806.11382*, 2018.

Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

Lillicrap, T. P. et al. Continuous control with deep reinforcement learning. *arXiv*, 2015.

Nilim, A. and El Ghaoui, L. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 2005.

Pinto, L. et al. Robust adversarial reinforcement learning. In *ICML*, 2017.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

Scherrer, B. and Geist, M. Local policy search in a convex space and conservative policy iteration as boosted policy search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014.

Tamar, A. et al. Scaling up robust mdps by reinforcement learning. *arXiv*, 2013.

Non-Parametric Off-Policy Policy Gradient

Samuele Tosatto
Intelligent Autonomous System
Technische Universität Darmstadt
64289 Darmstadt, Germany
samuele@robot-learning.de

Jan Peters*
Intelligent Autonomous System
Technische Universität Darmstadt
64289 Darmstadt, Germany
mail@jan-peters.net

Abstract

Policy gradients methods typically require a sample estimate of the state distribution induced by the policy, which results into excessive interaction with the environment after each policy update in order to avoid poor gradient estimation. In real applications, such as robotics, sample efficiency is a critical aspect.

Off-Policy policy gradient algorithms has been yet proposed in the literature, but they classically require the introduction of some strong approximations. For this reason, many works which relies on the off-policy policy gradient theorem, need to ensure the *behavioral policy* to be close to the *optimization policy*, eluding *de facto* the benefits provided by being off-policy.

We well define these source of approximations, and we propose an off-policy algorithm which does not rely on these approximations, leading to a better off-policy gradient estimation. We employ kernel regression and density estimation to obtain an approximation of both the value function and the state-distribution in closed form. This estimation directly yields an approximated analytical solution for the policy gradient. We show that the resulting policy gradient estimate is surprisingly accurate even with a fixed small amount of off-policy samples.

Keywords: policy gradient; non-parametric estimation; off-policy

Acknowledgements

This research has been funded by Bosch-Forschungsstiftung.

*Max Planck Institute for Intelligent Systems, 72024 Tübigen, Germany.

1 Introduction

Policy gradient estimation requires the computation of an integral w.r.t. the state distribution induced by the policy. A common choice is to approximate the integral by monte-carlo estimation, thus by sampling directly from the environment [14, 7, 10, 9]. When the policy is updated, the state-distribution changes, and new samples are required. This is the main cause of high sample complexity. The eventual decision to postpone the sampling, or to sample from another distribution, introduces an error in the estimation of the gradient. A common technique to mitigate this inefficiency is to introduce a divergence constraint between the estimated new distribution and the current sampled distribution [6, 9]. Since the state-distribution and the value-function can be computed in closed form for discrete Markov Decision Processes (MDPs) [13], we use non-parametric estimation, which allows us to work directly on samples.

2 State-of-the-art Off-Policy Policy Gradient

In the literature, there are various off-policy policy-gradient algorithms. Those algorithms are very often based on [1], which introduces three approximation in order to obtain the off-policy theorem: 1) *return modification*; 2) *gradient approximation*; and 3) *action detachment*.

Return modification. A gradient estimation is said to be off-policy if it can be computed by utilizing a state-action distribution different from the one induced from the policy. A first step to reach this goal is to introduce a new state-distribution ρ_β (usually the one induced by a *behavioral policy*) and by modifying the objective $J_{\pi,\beta} = \int_{\mathcal{X}} \rho_\beta(\mathbf{x}) V^\pi(\mathbf{x}) \, \mathrm{d}\mathbf{x}$ so that to maximize the expected return under ρ_β [11, 1]. It is very difficult to understand the meaning of the modified return in equation (2), since the value function depends on the policy π, while the state-distribution is generated by the behavioral policy. This objective has been inherited to all those algorithm which are developed upon the work from [11] or from [1], such as A3C [5], DDPG [4], TRPO [9] and PPO [10].

Gradient approximation. To avoid the dependency from the on-policy state-distribution [1] and [11] needed to introduce a further approximation in the computation of the gradient

$$\begin{aligned} \frac{\partial}{\partial\theta} J_{\pi,\theta} &= \int_{\mathcal{X}} \rho_\beta(\mathbf{x}) \int_{\mathcal{A}} \left(\frac{\partial}{\partial\theta} \pi_\theta(\mathbf{a}|\mathbf{x}) \right) Q^\pi(\mathbf{x},\mathbf{a}) + \pi_\theta(\mathbf{a}|\mathbf{x}) \left(\frac{\partial}{\partial\theta} Q^\pi(\mathbf{x},\mathbf{a}) \right) \mathrm{d}\mathbf{a} \, \mathrm{d}\mathbf{x} \\ &\approx \int_{\mathcal{X}} \rho_\beta(\mathbf{x}) \int_{\mathcal{A}} \left(\frac{\partial}{\partial\theta} \pi_\theta(\mathbf{a}|\mathbf{x}) \right) Q^\pi(\mathbf{x},\mathbf{a}) \, \mathrm{d}\mathbf{a} \, \mathrm{d}\mathbf{x}. \end{aligned}$$

The gradient obtained in this way does not depend on the state distribution μ_π induced by the optimization policy π. However this approximation introduces a relevant source of error.

Action detachment. In order to obtain an off-policy algorithm, being independent on the state distribution is not enough. One needs to be independent on how the actions are sampled. For the deterministic gradient the computation of $Q(\mathbf{x}, \pi_\theta(\mathbf{x}))$ there is no need to know how actions are sampled. On the contrary, the stochastic policy gradient needs to exploit the *log-ratio trick*, hence it requires the introduction of the *importance sampling* [2].

The first two approximations are coarse, and we think that this is the reason for which many methods introduce a KL constraint between the two policy distributions. By enforcing ρ_β to be close to μ_π, we are eluding the benefit of being off-policy.

3 Reinforcement Learning in Discrete World

In reinforcement learning's literature there is a special focus on the value function $\mathbf{v}_\pi$, which approximates the expected return for each observed state. On the contrary there is less focus on the dual counterpart of the value function, which is the state distribution[1] $\boldsymbol{\mu}_{\boldsymbol{\pi}i} = \sum_{t=0}^{\infty} \Pr(s_t = i | s_0, \pi)$ [8]. With $\boldsymbol{\mu}_{\boldsymbol{\pi}}$ we refer to the *discounted state visit count* d^π introduced in the work of [12] or the discounted version of N^π introduced by [13]. The value function and the state distribution can be defined as

$$\mathbf{v}_\pi = \mathbf{r}_\pi + \gamma P_\pi \mathbf{v}_\pi, \quad \boldsymbol{\mu}_{\boldsymbol{\pi}} = \boldsymbol{\mu}_0 + \gamma P_\pi^T \boldsymbol{\mu}_{\boldsymbol{\pi}} \tag{1}$$

where $\mathbf{r}_\pi$ is a vector of the mean reward, P_π is the transition matrix, and $\boldsymbol{\mu}_0$ is the initial state distribution. It is important to notice that equation on $\mathbf{v}_\pi$ is also referenced as Bellman equation, and it is extensively used also in the continuous world, since the integral $\int P(x'|x) V(x') \, \mathrm{d}x'$ is easy to approximate by sampling. On the contrary, the equation involving $\boldsymbol{\mu}_0$ is almost never used in the continuous world, since $\int P(x'|x) \mu(x) \, \mathrm{d}x$ is difficult to obtain. This difficulty is the main

[1] The term distribution is improper since ($\sum_i \boldsymbol{\mu}_{\boldsymbol{\pi}i} \neq 1$), but is just a matter of a multiplying factor.

reason why $\boldsymbol{\mu_\pi}$ is usually estimated on-policy (thus from the interaction of the policy with the environment). In the case of a discrete world, the equations in (1) are linear and can be solved in closed form

$$\mathbf{v}_\pi = \Lambda_\pi \mathbf{r}_\pi, \quad \boldsymbol{\mu_\pi} = \Lambda_\pi^T \boldsymbol{\mu}_0$$

where $\Lambda_\pi = (I - \gamma P_\pi)^{-1}$ is referenced as the *resolvent kernel*. The objective of reinforcement learning is usually to maximize a quantity called *return* $J_\pi = \boldsymbol{\mu}_0^T \mathbf{v}_\pi = \boldsymbol{\mu}_{\boldsymbol{\pi}}^T \mathbf{r}_\pi = \boldsymbol{\mu}_0^T \Lambda_\pi \mathbf{r}_\pi$ which estimates the average cumulative discounted reward.

Supposing that $\frac{\partial}{\partial \pi} P_\pi$ and $\frac{\partial}{\partial \pi} \mathbf{r}_\pi$ are known, we are able to obtain the policy gradient by

$$\begin{aligned} \frac{\partial}{\partial \pi} J_\pi &= \frac{\partial}{\partial \pi} \boldsymbol{\mu}_0^T \Lambda_\pi \mathbf{r}_\pi = \boldsymbol{\mu}_0^T \Lambda_\pi \left(\frac{\partial}{\partial \pi} \mathbf{r}_\pi \right) + \boldsymbol{\mu}_0^T \left(\frac{\partial}{\partial \pi} \Lambda_\pi \right) \mathbf{r}_\pi = \boldsymbol{\mu}_0^T \Lambda_\pi \left(\frac{\partial}{\partial \pi} \mathbf{r}_\pi \right) + \gamma \boldsymbol{\mu}_0^T \Lambda_\pi \left(\frac{\partial}{\partial \pi} P_\pi \right) \Lambda_\pi \mathbf{r}_\pi \\ &= \boldsymbol{\mu}_{\boldsymbol{\pi}}^T \left(\left(\frac{\partial}{\partial \pi} \mathbf{r}_\pi \right) + \gamma \left(\frac{\partial}{\partial \pi} P_\pi \right) \mathbf{v}_\pi \right). \end{aligned} \tag{2}$$

4 Non-Parametric Policy Gradient

We notice in the previous sections that in the discrete world is possible to obtain the state-distribution in closed form. However, its estimation in the continuous world is much more complex. Let us focus our attention to the continuous world view. We identify $V_\pi(\mathbf{x})$ as the value function evaluated in the state $\mathbf{x}$, $\mu_0(\mathbf{x})$ the initial state distribution, $r(\mathbf{x}, \mathbf{a})$ the average reward observed by the application of the action $\mathbf{a}$ in state $\mathbf{x}$, $p(\mathbf{x}'|\mathbf{x}, \mathbf{a})$ the state transition distribution. We define two different objectives, one for stochastic policies $\pi_\theta(\mathbf{a}|\mathbf{x})$ and one for deterministic ones $\pi_\theta(\mathbf{x})$.

Definition 1. *Objective with Stochastic Policy*

$$\begin{aligned} &\max_\theta J_\pi = \max_\theta \textstyle\int_{\mathcal{X}} V_\pi(\mathbf{x}) \mu_0(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &s.t. \, V_\pi(\mathbf{x}) = \textstyle\int_{\mathcal{A}} \Big(r(\mathbf{x}, \mathbf{a}) + \gamma \int_{\mathcal{X}} V(\mathbf{x}') p(\mathbf{x}'|\mathbf{x}, \mathbf{a}) \, \mathrm{d}\mathbf{x}' \Big) \pi_\theta(\mathbf{a}|\mathbf{x}) \, \mathrm{d}\mathbf{a} \quad \forall x \in \mathcal{X} \end{aligned}$$

Definition 2. *Objective with Deterministic Policy*

$$\begin{aligned} &\max_\theta J_\pi = \max_\theta \textstyle\int_{\mathcal{X}} V_\pi(\mathbf{x}) \mu_0(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &s.t. \, V_\pi(\mathbf{x}) = r(\mathbf{x}, \pi_\theta(\mathbf{x})) + \gamma \textstyle\int_{\mathcal{X}} V(\mathbf{x}') p(\mathbf{x}'|\mathbf{x}, \pi_\theta(\mathbf{x})) \, \mathrm{d}\mathbf{x}' \quad \forall x \in \mathcal{X} \end{aligned}$$

The constraints in definitions 1 and 2 are continuous in the sense that they are valid for $\forall \mathbf{x} \in \mathcal{X}$. This condition makes the problem intractable, and requires the introduction of an approximation. Very interestingly it is possible to solve the constraint in closed form following a non-parametric dynamic programming approach [3].

Non-Parametric Modelling. Let us assume to observe a set of n samples from the interaction with the system $D \equiv \{\mathbf{x}_i, \mathbf{a}_i, r_i, \mathbf{x}'_i\}_{i=1\dots n}$. Notice that there is no assumption on how the states $\mathbf{x}_i$ and the actions $\mathbf{a}_i$ are sampled, however the reward must be observed $r_i \sim r(\mathbf{x}_i, \mathbf{a}_i)$ from the interaction with the environment, as well as the next state $\mathbf{x}'_i \sim p(\cdot|\mathbf{x}, \mathbf{a})$. At this point, following the work proposed by [3], we introduce three kernel functions $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, $\varphi : \mathcal{A} \times \mathcal{A} \to \mathbb{R}^+$ and $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, which have to be symmetric, positive definite and must sum up to one (e.g., $\int_{s \in Sset} \psi(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x} \, \forall \mathbf{y} \in \mathcal{X}$). Let us define $\psi_i(\mathbf{x}) = \psi(\mathbf{x}, \mathbf{x}_i)$, $\varphi_i(\mathbf{a}) = \varphi(\mathbf{a}, \mathbf{a}_i)$ and $\phi_i(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{x}'_i)$. We approximate the reward function by using the Nadaraya-Watson regression and the transition with kernel density estimation

$$\tilde{r}(\mathbf{x}, \mathbf{a}) = \frac{\sum_{i=1}^n \psi_i(\mathbf{x}) \varphi_i(\mathbf{a}) r_i}{\sum_{i=1}^n \psi_i(\mathbf{x}) \varphi_i(\mathbf{a})}, \quad p(\mathbf{x}'|\mathbf{x}, \mathbf{a}) = \frac{p(\mathbf{x}', \mathbf{x}, \mathbf{a})}{p(\mathbf{x}, \mathbf{a})} \approx \frac{\sum_i \phi_i(\mathbf{x}) \psi_i(\mathbf{x}) \varphi_i(\mathbf{a})}{\sum_i \psi_i(\mathbf{x}) \varphi_i(\mathbf{a})} = \tilde{p}(\mathbf{x}'|\mathbf{x}, \mathbf{a}),$$

since $\tilde{p}(\mathbf{x}', \mathbf{x}, \mathbf{a}) = n^{-1} \sum_i \phi_i(\mathbf{x}) \psi_i(\mathbf{x}) \varphi_i(\mathbf{a})$ and $\tilde{p}(\mathbf{x}, \mathbf{a}) = n^{-1} \sum_i \psi_i(\mathbf{x}) \varphi_i(\mathbf{a})$. Following a similar reasoning as the one in [3] we can obtain a closed form solution for the value function, by using algebraic manipulation and the Galerkin's projection. Due to space constraint we omit the mathematical passages, but we recommend to the interested reader to read the work from [3] in order to have a rough idea. The approximated value function is

$$\tilde{V}_\pi(\mathbf{x}) = \kappa^T(x) \tilde{\Lambda}_\pi \Pi \mathrm{r} \tag{3}$$

where

$$\Pi_{i,j} = \begin{cases} \int_{\mathcal{A}} \varepsilon_j(\mathbf{x}_i, \mathbf{a}) \pi_\theta(\mathbf{a}|\mathbf{x}_i) \, \mathrm{d}\mathbf{a} & \text{for stoch. policy} \\ \varepsilon_j(\mathbf{x}_i, \pi_\theta(\mathbf{x}_i)) & \text{otherwise} \end{cases}, \quad P_{i,j} = \int_{\mathcal{X}} \phi_j(\mathbf{x}') \frac{\psi_i(\mathbf{x}')}{\sum_k \psi_k(\mathbf{x}')} \, \mathrm{d}\mathbf{x}', \quad \varepsilon_i(\mathbf{x}, \mathbf{a}) = \frac{\psi_i(\mathbf{x}) \varphi_i(\mathbf{a})}{\sum_j \psi_j(\mathbf{x}) \varphi_j(\mathbf{a})},$$

$\kappa_i(\mathbf{x}) = \psi_i(\mathbf{x}) / \sum_j \psi_j(\mathbf{x})$ and $\tilde{\Lambda}_\pi = (I - \gamma \Pi P)^{-1}$. We have solved the constraint, and we can now use the approximation of V_π obtained in order to compute the return and derive the gradient. Considering that

$$\tilde{J}_\pi = \int_{\mathcal{X}} \mu_0(\mathbf{x}) \kappa^T(\mathbf{x}) \tilde{\Lambda}_\pi \Pi \mathbf{r} \quad \text{and} \quad \frac{\partial}{\partial \theta} \tilde{\Lambda}_\pi = \gamma \tilde{\Lambda}_\pi \left(\frac{\partial}{\partial \theta} \Pi \right) P \tilde{\Lambda}_\pi \tag{4}$$

it is possible to compute the gradient w.r.t. J_π using our set of samples D.

Table 1: Experiment 1. Performance per dataset.

DATASET	α	$\dot{\alpha}$	ACTIONS	SAMPLES	NOPG-D	NOPG-S	DDPG	DDPG-Q
A	25	25	2	1250	-905± 434	-628± 263	-4344± 177	-3972± 135
B	27	27	2	1458	-480± 219	-593± 310	-4608± 116	-3946± 141
C	30	30	2	1800	-577 ± 345	-512± 232	-4310± 167	-4064± 132
D	40	40	2	3200	-487± 218	-371± 2.3	-4365± 189	-3966± 74

Experiment 1: Results obtained on three different datasets. The columns α, $\dot{\alpha}$ and ACTIONS contains the number of the discretization. Column SAMPLES refers to the number of samples contained in each dataset. The algorithm performance are evaluated over 50 runs.

Definition 3. *Gradient Approximated with Kernel Regression*

$$\frac{\partial}{\partial\theta}\tilde{J}_\pi = \int_{\mathcal{X}} \tilde{\mu}_\pi^T(\mathbf{x})\left(\frac{\partial}{\partial\theta}\Pi\right)\left(\mathbf{r}+\gamma\tilde{P}\tilde{\mathbf{v}}_\pi\right)\mathrm{d}\mathbf{x} \quad \textit{with} \quad \frac{\partial}{\partial\theta}\Pi_{i,j} = \begin{cases} \int_{\mathcal{A}} \varepsilon_j(\mathbf{x}_i,\mathbf{a})\pi_\theta(\mathbf{a}|\mathbf{x})\frac{\partial}{\partial\theta}\log\pi_\theta(\mathbf{a}|\mathbf{x}_j)\,\mathrm{d}\mathbf{a} \\ \frac{\partial}{\partial\mathbf{a}}\varepsilon_j(\mathbf{x}_i,\mathbf{a})|_{\mathbf{a}=\pi_\theta(\mathbf{x}_i)}\frac{\partial}{\partial\theta}\pi_\theta(\mathbf{x}_i) \end{cases} \tag{5}$$

and $\tilde{\mu}_\pi^T(\mathbf{x}) = \kappa^T(\mathbf{x})\tilde{\Lambda}\mu_0(\mathbf{x})$.

Notice, that differently from the known policy gradient methods, there is no need to approximate $\tilde{\mu}_\pi$ by collecting samples from the environment under the policy π On the contrary, it is possible to compute it in closed form given our set of off-policy samples D. Notice that all the remaining integrals can be approximated with Monte-Carlo sampling, and often one sample is enough to obtain a good estimation (we cannot discuss this point in detail due to space constraints).

5 Empirical Results

We tested NOPG in two different settings. The purpose of the first experiment is to test the ability of our algorithm to be truly off-policy, compared to DDPG (which is chosen here among off-policy algorithms for his sample efficiency). The second experiment aims to test the sample efficiency of NOPG.

5.1 Experiment 1: Off-Policy Analysis

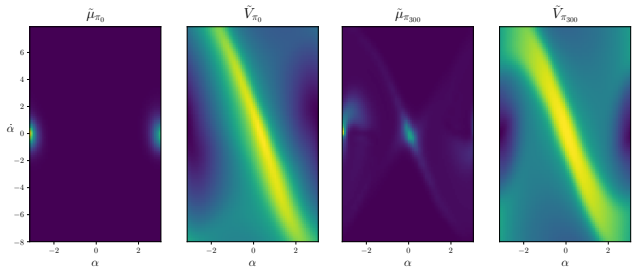


Figure 1: An example of the state-distribution and the value function estimated in the task of swing-up pendulum with our method. The plots shows the estimations before any policy improvement the optimal policy computed by NOPG-D algorithm (green correspond to higher values). Notice that the algorithm is able to predict that the system will reach the goal state $(\alpha = 0, \dot{\alpha} = 0)$ (third figure from the left).

The under-actuated swing-up pendulum is a classic task in reinforcement learning. We sampled the action-states using an uniform grid, and we observed for each state-actions pair the resulting reward and next state. We build four different size of dataset, with a different granularity of the grid. We compared the deterministic (NOPG-D) and the stochastic version (NOPG-S) of NOPG with DDPG, and DDPG-Q, which employs for the estimation of the Q-function the closed form solution obtained with NOPG. The idea of DDPG-Q is to provide a fairer comparison with NOPG. Table 1 summarizes the results. We want to note that our usage of DDPG and DDPG-Q on a fixed off-policy setting is improper, since DDPG prescribes a continuous interaction with the environment. In Figure 1 is possible to observe that the algorithm is able to recompute accurately the state distribution and the value function even if a uniform dataset was used.

5.2 Experiment 2: Sample Complexity

We selected the CartPole-v1 from the OpenAI Gym's library, in order test NOPG on a more complex system. In order to analyze the sample complexity we collected datasets of different sizes by the interaction of a random policy with the environment, until the desired number of samples is reached. The dataset is generate at the beginning of each run of the experiment using a random policy, so to obtain more variability.

In figure 2 we can observe the average number of steps obtained by the interaction of the policy found with NOPG. The maximal length of the episode is of 200 steps, but the episodes terminates before if physical limits are reached. every experiment is averaged on 50 seeds. Notice that since the dataset is generated with a random policy, the end of the track is never observed by the system, yielding an undeniable negative effect on the performances. We used in all our experiments a neural network of 50 neurons, with two output in the case of stochastic policy for encoding the mean and the log-standard-deviation of a Gaussian. We used Gaussian kernels for NOPG and for DDPG-Q.

5.3 Results

From the off-policy analysis in Section 5.1 is possible to argue that NOPG-D and NOPG-S are able to work in an off-line fashion (table 1), given a small fixed dataset obtained off-policy. Our technique is able to estimate accurately the state-distribution and the value function (figure 1), and to obtain a good policy-gradient estimation. On contrary, DDPG and DDPG-Q fail in finding a good solution.

Our sample complexity analysis carried out in Section 5.2 it is possible to argue that the proposed technique is sample efficient. The performance is negatively affected by the fact that in the generated dataset the limit of the cart are never reached.

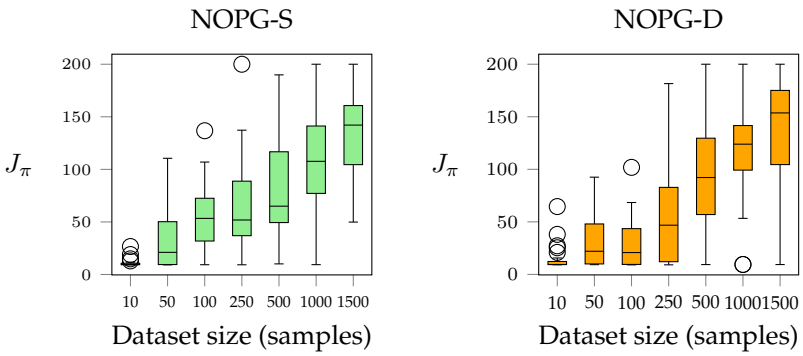


Figure 2: Performance of the proposed algorithms for the cart-pole environment. The return J_π indicates the average number of steps performed per episodes. It is possible to observe that NOPG-D generally is less efficient for a small number of samples, but it achieve higher results with a bigger datasets, in comparison to NOPG-S.

6 Conclusion and Future Work

We presented an off-line off-policy policy gradient algorithm, which overcomes the issues presented both for stochastic and deterministic policies. We tested our algorithm two environments, and achieve good solutions with minimal amount of samples. We also empirically showed that off-policy algorithms such as DDPG fail in converging to a good solution if fed with a fixed off-policy dataset. There is however vast space for improvement. The most critical aspect of NOPG is its poor scalability w.r.t. data. This aspect can be improved by introduce some sparsification in the matrix computation.

In conclusion, we have proposed a sample-efficient off-line policy gradient. This can be viewed as a possible starting point for developing more sample efficient on-line policy-gradient algorithms, which will potentially benefit the application of reinforcement learning directly to real system.

References

[1] T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[2] T. Jie and P. Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*, pages 1000–1008, 2010.

[3] O. B. Kroemer and J. R. Peters. A non-parametric approach to dynamic programming. In *Advances in Neural Information Processing Systems*, pages 1719–1727, 2011.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[6] J. Peters, K. Mülling, and Y. Altun. Relative entropy policy search. In *AAAI*, pages 1607–1612. Atlanta, 2010.

[7] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[8] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[11] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[12] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[13] X. Wang and T. G. Dietterich. Model-based policy gradient reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 776–783, 2003.

[14] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Forgetting Process in Model-Free and Model-Based Reinforcement Learning

Asako Toyama
Department of Cognitive and Psychological Sciences
Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Japan
asako.toyama@gmail.com

Kentaro Katahira
Department of Cognitive and Psychological Sciences
Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Japan
katahira.kentaro@b.mbox.nagoya-u.ac.jp

Hideki Ohira
Department of Cognitive and Psychological Sciences
Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Japan
ohhira.hideki@a.mbox.nagoya-u.ac.jp

Abstract

In commonly used standard reinforcement learning models, values are updated only for the chosen options while the values remain unchanged for the other options. On the other hand, when applying reinforcement learning models to animals and humans, it is more natural to assume that the learned values are lost over time as a consequence of memory decay (i.e., the forgetting process). Thus, we compared a standard reinforcement learning model to a reinforcement learning model that includes a forgetting process, using human choice data from a two-stage decision task in which the reward probability changed slowly and randomly over the trials. The algorithm used to implement the forgetting process was similar to that of the learning process. In the learning process, the values of the chosen options were assumed to be updated toward the obtained outcome and a learning rate adjusts the degree of updating. On the other hand, in the forgetting process, the values of the unchosen options were assumed to gradually approach towards a default value, which is a new concept introduced in our model, and it was also assumed that a forgetting rate adjusts the degree of change. The data were well fitted by including the forgetting process. Moreover, our simulation data demonstrated possible estimation biases due to fitting data using a model without the forgetting process.

Keywords: Reinforcement Learning; Model-Based; Forgetting Process; Default Value; Estimation Bias

Acknowledgements

This study was supported by Grant-in-Aid for Early-Career Scientists (18K13366 to AT), by Grant-in-Aid for Scientific Research (B) (18KT0021 to KK) and by Grant-in-Aid for Scientific Research (B) (17H02649 to HO).

1 Introduction

In standard reinforcement learning (RL) models, values are updated only for the chosen options while the values remain unchanged for the other options. On the other hand, when applying RL models to animals and humans, it is more natural to assume that the early-learned values are lost over time as a consequence of memory decay (i.e., the forgetting process). It has been reported that the inclusion of the forgetting process in RL models improved fits to actual choice data obtained from rats (Ito & Doya, 2009), monkeys (Barraclough, Conroy, & Lee, 2004), and humans (Niv et al., 2015). In these models, the values of the unchosen options were assumed to gradually approach zero. This assumption may be proper when the calculated values represent the association strength between action and outcome. However, when the values express expected values rather than association strengths, it is possible to encounter a case in which unchosen options increase their values because of their increased uncertainty. Thus, we introduce a new concept, *default value*, which represents the endpoint that the values of the unchosen options gradually approach (Toyama, Katahira, & Ohira, 2017). It also corresponds to an expected value for options in the absence of experience or knowledge regarding the relationship between option and outcome. In this study, we examine the validity of the forgetting process in an RL model using experimental data. For this purpose, we used a two-stage decision-making task developed by Daw, Gershman, Seymour, Dayan, and Dolan (2011) and their computational model which provides a parameter representing the balance between model-based and model-free learning. We also simulated possible biases in estimating this parameter from the data including the forgetting process by using a model which does not include the forgetting process.

2 Methods

In the experiment, 23 students participated. The same task was used in both the experiment and the simulations.

1.1 Task

Participants conducted a two-stage decision-making task including 303 trials (Figure 1). In each trial, they were required to choose one of two options at the first and second stages successively and the second-stage choice produced a feedback (rewarded or unrewarded). Importantly, each option at the first stage has a propensity to transition to a particular state of the second stage. That is, an option in the first stage leads to one of the second-stage states in 70% cases and the other in 30% cases, whereas the other option leads to these states in a reversed manner. Thus, different choice preferences can be predicted between when participants use a transition model or not. The reward probabilities of each second-stage option changed slowly and in random directions over the trials, and the same reward probabilities were used for all participants.

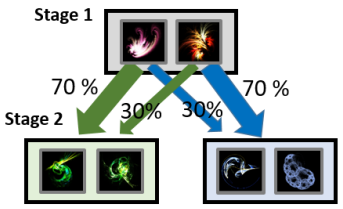


Figure 1. The behavioral task

1.2 Computational model

We used a model proposed by Daw et al. (2011). The model-free system uses SARSA(λ) TD learning (Rummery & Niranjan, 1994) to update state-action values, $Q_{MF}(s_i, a_i)$, at each stage i in each trial. The action values chosen at the first stage, $Q_{MF}(s_1, a_1)$, and at the second stage, $Q_{MF}(s_2, a_2)$, are updated, respectively, by the state prediction error and the reward prediction error (RPE) as follows:

$$Q_{MF}(s_1, a_1) \leftarrow Q_{MF}(s_1, a_1) + \alpha_{L1}\big(Q_{MF}(s_2, a_2) - Q_{MF}(s_1, a_1)\big), \quad (1)$$

$$Q_{MF}(s_2, a_2) \leftarrow Q_{MF}(s_2, a_2) + \alpha_{L2}\big(r_2 - Q_{MF}(s_2, a_2)\big), \quad (2)$$

where α_{L1} and α_{L2} are the learning-rate parameters of each stage and r_2 denotes the reward (1 or 0). The first-stage value is again updated by the second-stage RPE but its impact is discounted as follows,

$$Q_{MF}(s_1, a_1) \leftarrow Q_{MF}(s_1, a_1) + \alpha_{L1}\lambda\big(r_2 - Q_{MF}(s_2, a_2)\big), \tag{3}$$

where λ is an eligibility-trace decay parameter. Regarding the second stage, Q_{MF} calculated with Eq. (2) is equal to the model-based value, Q_{MB}, because there is no transition to a further stage. On the other hand, the model-based values of the first-stage actions ($a_k \in a_{1,1}, a_{1,2}$) are computed anew each time as follows:

$$Q_{MB}(s_1, a_k) = \sum_{\acute{s}} T(\acute{s}|s_1, a_k) \max_{\boldsymbol{a}} Q_{MB}(\acute{s}, \boldsymbol{a}), \tag{4}$$

where $T(\acute{s}|s_1, a_k)$ is a transition-probability function that describes the probability of moving to a second-stage state $\acute{s}$ after choosing action a_k at the first stage. $\boldsymbol{a}$ is the set of possible actions in state $\acute{s}$. $T(\acute{s}|s_1, a_k)$ was simply defined as a value of either 0.7 or 0.3 depending on the experienced transition frequency. Finally, the first-stage Q_{MF} and Q_{MB} are integrated into a net value, Q_{NET}, with a weighting parameter $0 \le w \le 1$.

$$Q_{NET}(s_1, a_k) = wQ_{MB}(s_1, a_k) + (1 - w)Q_{MF}(s_1, a_k). \tag{5}$$

For action selection, the net values are converted to probabilities of choosing each action using a softmax function where the degree of value-dependency and the tendency towards value-independent perseveration are adjusted respectively by the inverse temperature, β_1 and β_2, and the choice trace weight, ϕ.

Forgetting process: To implement the forgetting process in the above model, we introduced a forgetting parameter, $0 \le \alpha_F \le 1$, in each stage ($\alpha_{F1}$and α_{F2}). The values of unchosen actions are updated as follows:

$$Q_{MF}(\tilde{s}_i, \tilde{a}_i) \leftarrow Q_{MF}(\tilde{s}_i, \tilde{a}_i) + \alpha_F\big(\mu - Q_{MF}(\tilde{s}_i, \tilde{a}_i)\big), \tag{6}$$

where $Q_{MF}(\tilde{s}_i, \tilde{a}_i)$ represents the values of the unchosen actions and the values of the unvisited-state actions in stage *i*. The parameter $0 \le \mu \le 1$ is a *default value* to which unchosen action values are regressed. Under this rule, all initial Q_{MF} values are set to equal μ. Hereafter, the model including the forgetting process is called *the model with forgetting* and the model excluding this process is called *the model without forgetting*.

3 Experimental results

Parameter estimation and model comparison: The parameters were estimated for each participant by a maximum log-likelihood method. For a comparison of the models, we computed the Bayesian information criterion (BIC; Schwarz, 1978) for each participant in each model as BIC = -2*LL* + *k* log(*n*), where *LL* is the log likelihood of the choice probabilities, *k* is the number of free parameters, and *n* is the total number of choices. The model with forgetting was significantly favored (with smaller BIC values) over the model without forgetting [$t(22) = 3.33$, $p = .003$, $d = 0.69$]. Of the 23 participants, 17 supported the model with forgetting, as shown in Figure 2A, which indicates the differences in BIC between the two models for each participant.

Default value parameter: From Eq. (6), it is predictable that a high default value μ increases the values of the unchosen options. Therefore, a high μ will capture a type of exploration behavior or a choice shift caused by an expectation for recently unchosen uncertain options. To confirm this prediction, we conducted Pearson correlation analysis between the estimated μ and the average stay probabilities at the first stage. There was a negative correlation ($r = -.56$, $p = .005$; Figure 2B). In other words, those who had a higher default value or a higher expectation for unchosen options showed greater exploration behavior.

Forgetting process and estimation of parameter *w*: Figure 2C shows the correspondence of the estimated model-based weighting parameters w of the two models. The values of w for 20 of 23 participants were estimated to be lower when estimated by the model without forgetting than the model with forgetting. The degree of difference was different among the participants (minimum = 0.52, mean = -0.11, maximum = -0.88).

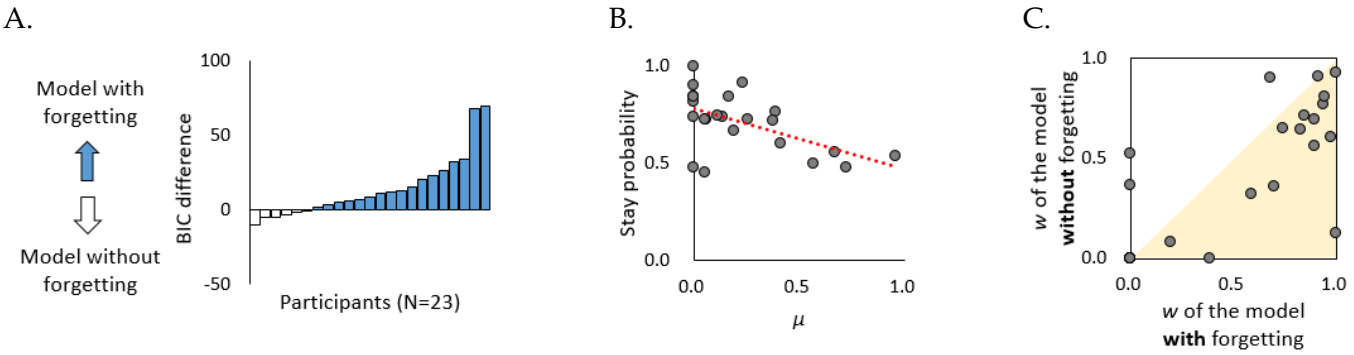


Figure 2. (A) Differences between the model with forgetting and the model without forgetting in the Bayesian information criterion (BIC) scores for each participant. (B) Relationship between the average stay probabilities of all trials at the first stage and the parameter μ estimated by using the model with forgetting. (C) Relationship between the parameter w estimated by the model without forgetting and w estimated by the model with forgetting.

4 Simulations

Next, we examined possible parameter estimation biases, especially the biases in the model-based weight parameter, when data is fitted using a model without forgetting. For this purpose, synthetic datasets with various forgetting rates and model-based weights were prepared. These were then fitted for the model with forgetting and the model without forgetting.

Synthetic data: Synthetic datasets for four simulations were generated from the models with forgetting. Simulations 1 and 2 tested the effect of the first-stage forgetting rate α_{F1}, and simulations 3 and 4 tested the effect of the second-stage forgetting rate α_{F2}. These parameters were varied from 0 to 0.3 in steps of 0.05. The parameter w was also changed from 0 to 1 in steps of 0.2. Thus, there were 42 true parameter sets in each simulation. Considering the direction of value changes in the forgetting process, we examined two cases: $\mu = 0$ (simulations 1 and 3) and $\mu = 1$ (simulations 2 and 4). The other parameters were fixed ($\alpha_{L1} = 0.2, \alpha_{L2} = 0.3, \beta_1 = 7, \beta_2 = 7, \lambda = 0.4, \phi = 0.8$ and $\alpha_{F1} = 0.2$ or $\alpha_{F2} = 0.3$). From each true parameter set, the choice data of 100 agents performing the two-stage task with 303 trials were produced.

Fitting by the models with and without forgetting: The generated data were fitted with the model including the full set of free parameters (the full model) or with the model in which the target parameter, α_{F1} or α_{F2}, was set to zero, and the parameter μ was fixed at the true value. Figure 3 shows the estimated values of w when the data were fitted with the full model (solid blue line), or the model that lacked the forgetting process in either the first stage (simulations 1 and 2, solid red line) or the second stage (simulations 3 and 4, solid red line). Regardless of the value of μ, if the model lacked the first-stage forgetting process, the larger the true α_{F1} was, the more the estimated w was biased in the model-based direction (simulations 1 and 2). In contrast to this, when the model lacked the second-stage forgetting process, the larger the true α_{F2} was, the more the estimated w was biased in the model-free direction (simulations 3 and 4).

To consider the possible causes of the observed biases, we focused on the first-stage model-free and model-based values because the parameter w only has meaning in this stage. From the equations of the computational model, it is clear that parameter α_{F1} relates to the calculation of the first-stage model-free values but not the model-based values, whereas the parameter α_{F2} relates to the calculation of the first-stage model-based values but not the first-stage model-free values. In general, the parameter estimation is biased to weight the more accurate part of the calculation to enhance the fit to the data. Hence, the parameter w is biased in the model-based direction when the fitting model lacks α_{F1} (simulations 1 and 2) and in the model-free direction when the fitting model lacks α_{F2} (simulations 3 and 4). In addition, our simulations showed that the larger the true forgetting rate was, the more the estimation of w was biased.

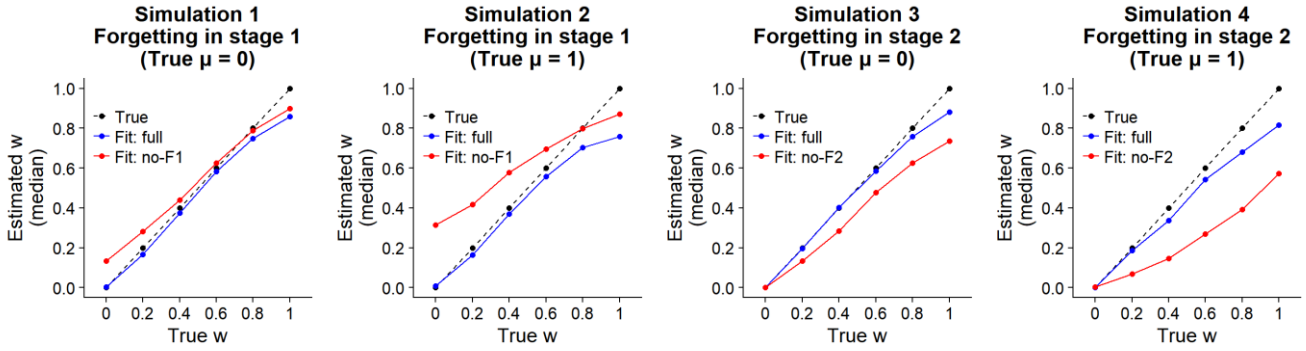


Figure 3. Estimation bias for the parameter w. The synthetic data for simulations 1 and 3 were generated by the model with forgetting in which μ was set to zero, and those for simulations 2 and 4 were generated by the model with forgetting in which μ was set to one. The panels show the true w in the generating model (dashed black line), the median of the estimated w according to the full model (solid blue line), and the median of the estimated w according to the model without forgetting (no-F1 mode is $\alpha_{F1} = 0$ and no-F2 mode is $\alpha_{F2} = 0$; solid red line).

5 Conclusion

In the experimental study, the inclusion of the forgetting process in the model lead to an improvement in the model fit. Considering that many studies have demonstrated that uncertainty prompts exploration behavior (Badre, Doll, Long, & Frank, 2012), it is supposed that recently unchosen options might carry some vague expectation related to their increased uncertainty. This may be captured by the default-value parameter introduced in our model which showed a positive correlation with the exploration behavior. However, most of the previous studies used models without forgetting. Model misspecification is known to cause undesirable biases in parameter estimation (Katahira, 2018). In our simulations, it was revealed that the weighting parameter w was biased towards the model-based direction when the fitting model lacked the first-stage forgetting process. However, it was biased towards the model-free direction when the fitting model lacked the second-stage forgetting process. Considering the fact that the estimations from the experimental data were biased to the model-free direction, the forgetting process in the second stage is more critical in this task. This is reasonable because the second-stage has more options than the first-stage and the effect of the forgetting process may be more severe than at the first-stage. In summary, our study proposes the inclusion of the forgetting process into RL models and also shows the possible estimation biases arising from the use of a model without forgetting.

6 References

Badre, D., Doll, B. B., Long, N. M., & Frank, M. J. (2012). Rostrolateral prefrontal cortex and individual differences in uncertainty-driven exploration. *Neuron, 73*(3), 595-607.

Barraclough, D. J., Conroy, M. L., & Lee, D. (2004). Prefrontal cortex and decision making in a mixed-strategy game. *Nature Neuroscience, 7*(4), 404-410.

Daw, N. D., Gershman, S. J., Seymour, B., Dayan, P., & Dolan, R. J. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron, 69*(6), 1204-1215.

Ito, M., & Doya, K. (2009). Validation of decision-making models and analysis of decision variables in the rat basal ganglia. *Journal of Neuroscience, 29*(31), 9861-9874. 9

Katahira, K. (2018). The statistical structures of reinforcement learning with asymmetric value updates. *Journal of Mathematical Psychology, 87*, 31-45.

Niv, Y., Daniel, R., Geana, A., Gershman, S. J., Leong, Y. C., Radulescu, A., & Wilson, R. C. (2015). Reinforcement learning in multidimensional environments relies on attention mechanisms. *Journal of Neuroscience, 35*(21), 8145-8157.

Rummery, G., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems. (Technical Report CUED/F-INFENG/TR 166).* Cambridge: Cambridge University.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics, 6*(2), 461-464.

Toyama, A., Katahira, K., & Ohira, H. (2017). A simple computational algorithm of model-based choice preference. *Cognitive, Affective & Behavioral Neuroscience*, 17(4), 764-783.

A Human-Centered Approach to Interactive Machine Learning

Kory W. Mathewson*
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
korymath@gmail.com

Abstract

The interactive machine learning (IML) community aims to augment humans' ability to learn and make decisions over time through the development of automated decision-making systems. This interaction represents a collaboration between multiple intelligent systems—humans and machines. A lack of appropriate consideration for the humans involved can lead to problematic system behaviour, and issues of fairness, accountability, and transparency. This work presents a human-centred thinking approach to applying IML methods. This guide is intended to be used by AI practitioners who incorporate human factors in their work. These practitioners are responsible for the health, safety, and well-being of interacting humans. An obligation of responsibility for public interaction means acting with integrity, honesty, fairness, and abiding by applicable legal statutes. With these values and principles in mind, we as a research community can better achieve the collective goal of augmenting human ability. This practical guide aims to support many of the responsible decisions necessary throughout iterative design, development, and dissemination of IML systems.

Keywords: human-in-the-loop, applied interactive machine learning, responsible machine learning, fairness, accountability, transparency, human-centred design

Acknowledgements

This research was partially supported by the University of Alberta, Alberta Machine Intelligence Institute (Amii), the Natural Sciences and Engineering Research Council (NSERC), and the Alberta Innovates Technology Fund (AITF). I am grateful to Dr. Patrick Pilarski, Dr. Matt Taylor, Alex Kearney, Johannes Günther, Jamie Brew, Lana Cuthbertson, Katya Kudashkina, Keyfer Mathewson, and Dr. Piotr Mirowski for sharing their comments during the preparation of this manuscript. Any errors are my own.

*https://korymathewson.com

1 Introduction and Background

Machine learning (ML) comprises a set of computing science techniques for automating knowledge acquisition rather than relying on explicit human instruction. Interactive ML (IML) systems incorporate humans with ML techniques. The goals of IML include the amplification and augmentation of human abilities. All ML system have humans at some point in the learning loop; some interactions are more evident than others. For example, interacting humans may provide data, objective functions, direct feedback, algorithms, or code. Some system designers choose to acknowledge and feel a responsibility for these humans more than others.

Examining ML from a human-centred perspective requires re-framing work-flows to include humans during conceptualization, implementation, and interaction. Human-centred thinking will lead to more usable ML systems, improved mutual understanding, and augmented communication between intelligent systems [18]. These benefits come at the increased cost of time and energy to understand human factors and many designers choose to ignore social factors. Ignorance has serious implications (i.e. reward misalignment between humans and machines [6, 2, 22, 12].

Human-centred design develops solutions to problems by involving the human perspective in all steps of the process. Human-centred systems put people before machines, and delight in the ability and ingenuity of humans [3].

Fails and Olsen (2003) [5] introduced interactive machine learning (IML). Amershi *et al.* (2014) [1] discuss interaction modalities, and the AAAI-17 tutorial, "Interactive Machine Learning: From Classifiers to Robotics", presented a comprehensive history of the field [9]. Incorporating human decision making with ML systems continues to be an active area of research [15, 8, 12, 13]. Previous work has discussed the virtues of human-centered thinking in ML [7, 14]. But, there continues to be a gap between the mental models of responsible ML practitioners and many contemporary ML deployments.

A good proportion of ML research happens at institutions with ethics review boards. These groups should rigorously apply ethical principles through review of proposed studies. But, they may not exist in your organization, or they may not provide a comprehensive review. This guide details many of the ethics review board questions and extends beyond these considerations. Many ML systems will find their way out of laboratory settings without ethics review. For instance, many ML studies will have no human subjects, but this research (i.e. the code, data, and models) will be released and used by the general public. Design, development, and deployment can happen without subscribing human subjects directly.

Sec. 2 focuses on the humans-in-the-loop, the goals and hypotheses of the research, and the data. This section also presents two design thinking exercises: a *whiteboard model*–which can help identify streamlined processes, and a *premortem*–which can help identify potential risks and failure modes early in the process. Sec. 3 concerns itself with the the iterative development process. It advocates for checking in with the humans-in-the-loop at each iteration. Sec. 4 focuses on questions of model deployment and public communication. It can be helpful to read through this guide before starting the project and to continually check-in with it for design, development, and dissemination of human-interactive decision-making systems.

2 Human-Centered Design

Step 1: Define the hypothesis State the investigated question of interest. Can you pose it as a testable hypothesis [20], which can be supported by evidence? The premise is the starting point and motivation for investigation.

Step 2: Loop in humans Define your values and principles. How will you align this work with human needs? Identify individual and societal social factors in the problem. Start by considering why your hypothesis is essential not just to you, but to the larger community of people who might interact with it. How will the system augment human lives?

Consider the question of interest from multiple stakeholder perspectives. For instance, think about three groups of individuals: those invested in the success of the work, those impacted by your work, and those who might be interested in the work. Empathize with stakeholders to appreciate how the problem, and potential solutions, will affect them.

Consent starts with communication. Build open-communication channels with stakeholders in these groups. Gather from them ideas, design requirements, concerns, and questions. You should review your values alongside the values of these individuals. How will these stakeholders engage with your system? Thinking about this now will help during Step 9: Deploy. How might the interaction look? Ask each of them how they will evaluate the performance of your system? Discuss how the system might be used both constructively and misapplied to harm. This dual-use discussion is ongoing in the field of ML [21].

Choices you make will impact these people directly, and you are responsible for the impact of your work on them. Embrace this responsibility. These stakeholders can champion your system if you engage them early in the process and often through iteration.

Step 3: Define the goal Define a specific, measurable, attainable, relevant, and timely goal. It should be linked directly to the hypothesis from Step 1: Hypothesis. The goal should clearly define success. This definition will embrace the ways that your stakeholders will engage with, and evaluate the performance of your system.

Often there are multiple metrics which define success for a given problem. ML system designers often refine optimization to a single metric of interest. Consider both your optimization metric (i.e. model performance indicator) and your measures of system success. How does this learning objective align with the ways that your stakeholders will evaluate your system? Define a testing suite for safety which you will use in Step 6: Evaluate your model. These tests should consider human health, safety, and well-being. As well, they should evaluate your system on biases, fairness, and equality across hidden features in your data [11].

It can help to align your work with familiar categories of existing ML work [10]. Is this project developing a new model, applying existing methods to new data, or presenting a new model of human behaviour? Does the system test the limitations of current models on a new problem?

Given your goal, what are the technical, scientific, implementation problems which need to be solved? You should be able to break your system into small components (e.g. data, processing, evaluation). Doing so will make addressing each part individually easier. What are some of the downstream impacts of accomplishing your goal? That is, if the results support your hypothesis, what else might be true?

Step 4: Define the data An ML system is a reflection of the training data it learns on. It can reflect many common human biases. What is your ideal data set? How much data do you desire? How much data do you need? Why might these amounts be different? What are the dependent and independent variables? How will the data be organized and represented for the learning system? How might possible data sources stray from the ideal ort data? How will you define what outliers, and bad data points, are?

How will you accumulate, clean, parse, label, and safely store your data? How might you fill in blanks in your data? Can you use software to simulate data? Experiments on simulated data designed to test assumptions and gain intuition provide valuable insights. How will you incorporate new data which arrives after deployment?

How will you handle participant recruitment and compensation? If you pay for data (e.g. through crowd-sourcing, direct payment to humans, or a third party), what are the costs of accumulating data? What are the usage rights and responsibilities of your data? What is the ownership model for this data?

If you have humans in your data collection, consider the ethical implications of collecting their data. How are your data generating humans informed of the use of their data? How are data privacy and security communicated? What are the potential biases and sensitivities in human-collected data (i.e. personal or identifying information)?

Once you collect your data, split it into training, evaluation (i.e. validation), and held-out testing data segments. Do this early, lest you leak information from the test segment into your model selection and parameter tuning processes.

2.1 Design Thinking Exercises

The Whiteboard Model For the whiteboard model exercise consider the following: given your hypothesis, stakeholder analysis, and goal, how would you get to a solution given a short amount of time and only a whiteboard? It is tempting to think about novel techniques which might address your goal. Preferably, it is often more effective to make something that works (i.e. your whiteboard model) and then make iterative improvements. This thought exercise will also provide an opportunity to mentally zoom-out from the problem and think about how potential solutions fit into a broader direction.

The Pre-mortem For the pre-mortem exercise, imagine that the project fails for a variety of reasons. Write down these failure modes. Then, for each failure mode, work backwards to identify what might have lead to different results. This process of prospective hindsight can increase the ability to identify reasons for future outcomes by 30% [17] correctly. A pre-mortem can provide insights and ideas which you can use in the next iterative development steps and can help reduce the chances of arriving at predictable failure modes.

3 Develop, Analyze, Evaluate, and Iterate

Step 5: Build model Safe design is the first step towards safe use. Think about model misuse starting with the first model you build. Step 2: Loop in Humans covered much of this preparation.

Consider simple models for your learning from your data. A simple model serves as a baseline for comparing model improvements. What is your baseline model? It might be a model that generates random outputs; *random* is a perfectly reasonable baseline and can help to identify other bugs in the development pipeline. Other reasonable benchmarks

include a 'majority-class' model that predicts the most common output in the training set and a 'by-hand' model which invites a human to consider the inputs and generate an output.

The 'by-hand' model is often called a Wizard-of-Oz, or human-assisted model, and has been used at scale by large tech companies to help understand human interactions [16]. Similar to the *whiteboard model*, these baselines will help to define essential features in your data for the given performance metric.

Step 6: Evaluate model Your evaluation data will serve as a consistent comparison for model improvement. Test your model on your evaluation data segment. Track your key metric. The performance of your baseline model starts as your 'best,' and 'worst,' performing model. Keep your model performances as comparators as you iterate in Step 8: Re-evaluate and Iterate. What the limitations of your evaluation scheme? What are the unaccounted costs or errors? How does the model perform on the evaluation data and the safety suite designed in Step 3: Define the goal.

With each model evaluation iteration, it is essential to think about biases, fairness, and equality across diverse groups. Each iteration is a crucial checkpoint to communicate with stakeholders. Your stakeholders' discussions should include how they feel your model has addressed the ideas, interests, design requirements, concerns, and questions brought up in Step 2: Loop in Humans. How do they evaluate system performance? How would the baseline model impact them? Consider your stakeholders might be satisfied with your baseline model.

Step 7: Analyze trade-offs You will make trade-offs as you make iterate models, and model parameters, towards optimizing your crucial metric. Consider these trade-offs by listing each of them and their associated impacts independently. Trade-offs often include factors such as cost, storage, learning speed, inference speed, computation complexity, model serving, deployment, and human interpretability. It helps to perform ablation studies which systematically remove model components to determine their relative contributions. Considering each of these trade-offs will help you iterate on your model development.

Step 8: Re-evaluate and iterate Given the trade-offs defined in Step 7, review your key metrics. Ensure you capture all the information required before continuing. For instance, how do you log experimental parameters (i.e. model information) and results? Once you are confident that you can systematically make model improvements towards your evaluation metric, then it is time to iterate through Steps 5, 6, and 7. Once your evaluation performance converges, only then should you test your model(s) on the held out test set data segment. This testing should be used to compare models, and not to tune model parameters.

4 Disseminate

Step 9: Deploy the system Present and test the system with your stakeholders and individuals you have not engaged with up to this point. When testing with humans, focus on usability. How are stakeholders interacting with your model? Usability can have a profound effect on the perceived quality and capability of models. These are valuable interactions, note how these humans interpret the performance of your system.

Consider that many humans may act against the system, by accident or on purpose. How will you handle attacks on your model? What are the technical security implications of model security and the value-based human-experience design choices you have made which may influence human-behaviour? What are the fail-safes and procedural safeguards and how can you adapt them during deployment? How are you communicating the risks of interaction?

Step 10: Communicate The purpose of communication is to convey the key ideas to your audience clearly so that they may comprehend them with minimal effort. You should be able to state your key results and how it aligns with your hypothesis? Do your results match or contradict similar work? What are the limitations of the current model? How might these problems be addressed in the future? Do the results challenge any of the ideas or beliefs of the stakeholders?

When communicating the project, it is helpful to follow the ML Reproducibility checklist [19]. Can you open source your code, data, models, and deployment? Consider how and why others might attempt reproduction.

5 Conclusions

Human-centred thinking for design and development can lead to substantial improvements in the development and adoption processes. By empathizing with those invested in, impacted by, or adversaries of ML systems, developers can better serve the needs of all humans involved. Enabling human to efficiently and effectively interact with systems continues to be a key design challenge [4]. Human-centric design in ML can help address ongoing challenges of bias and unfairness and potentially improve the transparency and accountability of the choices which go into designing, developing and deploying new systems.

References

[1] Saleema Amershi et al. "Power to the people: The role of humans in interactive machine learning". In: *AI Magazine* 35.4 (2014), pp. 105–120.

[2] Danton S Char, Nigam H Shah, and David Magnus. "Implementing machine learning in health care - addressing ethical challenges". In: *The New England journal of medicine* 378.11 (2018), p. 981.

[3] Mike Cooley. "On human-machine symbiosis". In: *Human Machine Symbiosis*. Springer, 1996, pp. 69–100.

[4] John J Dudley and Per Ola Kristensson. "A Review of User Interface Design for Interactive Machine Learning". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 8.2 (2018), p. 8.

[5] Jerry Alan Fails and Dan R. Olsen Jr. "Interactive Machine Learning". In: *Proceedings of the 8th International Conference on Intelligent User Interfaces*. IUI '03. Miami, Florida, USA: ACM, 2003, pp. 39–45. ISBN: 1-58113-586-6.

[6] *Fairness, Accountability, and Transparency in Machine Learning*. http://www.fatml.org/. 2019-02-26.

[7] *Human Centred Machine Learning CHI2016*. http://hcml2016.goldsmithsdigital.com/. 2019-02-26.

[8] Borja Ibarz et al. "Reward learning from human preferences and demonstrations in Atari". In: *CoRR* abs/1811.06521 (2018). arXiv: 1811.06521.

[9] *Interactive Machine Learning: From Classifiers to Robotics*. https://eecs.wsu.edu/~taylorm/17AAAITutorial.html. 2019-02-26.

[10] Pat Langley. "Crafting papers on machine learning". In: *ICML*. 2000, pp. 1207–1216.

[11] Jan Leike et al. "AI Safety Gridworlds". In: *CoRR* abs/1711.09883 (2017). arXiv: 1711.09883.

[12] Jan Leike et al. "Scalable agent alignment via reward modeling: a research direction". In: *arXiv preprint arXiv:1811.07871* (2018).

[13] Zhiyu Lin et al. "Explore, Exploit or Listen: Combining Human Feedback and Policy Model to Speed up Deep Reinforcement Learning in 3D Worlds". In: *CoRR* abs/1709.03969 (2017). arXiv: 1709.03969.

[14] Josh Lovejoy and Jess Holbrook. *Human-Centered Machine Learning*. https://medium.com/google-design/human-centered-machine-learning-a770d10562cd. 2019-02-26.

[15] Kory Wallace Mathewson and Patrick M. Pilarski. "Simultaneous Control and Human Feedback in the Training of a Robotic Agent with Actor-Critic Reinforcement Learning". In: *CoRR* abs/1606.06979 (2016). arXiv: 1606.06979.

[16] Cade Metz. *Facebook's Human-powered Assistant May Just Supercharge AI*. https://www.wired.com/2015/08/how-facebook-m-works/. 2019-02-26.

[17] Deborah J Mitchell, J Edward Russo, and Nancy Pennington. "Back to the future: Temporal perspective in the explanation of events". In: *Journal of Behavioral Decision Making* 2.1 (1989), pp. 25–38.

[18] Patrick M Pilarski et al. "Communicative Capital for Prosthetic Agents". In: *arXiv preprint arXiv:1711.03676* (2017).

[19] Joelle Pineau. *The Machine Learning Reproducibility Checklist*. https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf. 2019-02-26.

[20] Karl Popper. *The logic of scientific discovery*. Routledge, 2005.

[21] *Some thoughts on zero-day threats in AI, and OpenAI's GP2*. https://fast.ai/2019/02/15/openai-gp2. 2019-02-26.

[22] Muhammad Bilal Zafar et al. "Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment". In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2017, pp. 1171–1180.

Thompson Sampling for a Fatigue-aware Online Recommendation System

Yunjuan Wang
Electrical and Computer Engineering
University of Illinois at Chicago
ywang581@uic.edu

Theja Tulabandhula*
Information and Decision Sciences
University of Illinois at Chicago
tt@theja.org

Abstract

In this paper we consider an online recommendation setting, where a platform recommends a sequence of items to its users at every time period. The users respond by selecting one of the items recommended or abandon the platform due to fatigue from seeing less useful items. Assuming a parametric stochastic model of user behavior, which captures positional effects of these items as well as the abandoning behavior of users, the platform's goal is to recommend sequences of items that are competitive to the single best sequence of items in hindsight, without knowing the true user model a priori. Naively applying a stochastic bandit algorithm in this setting leads to an exponential dependence on the number of items. We propose a new Thompson sampling based algorithm with expected regret that is polynomial in the number of items in this combinatorial setting, and performs extremely well in practice. We also show a contextual version of our solution.

Keywords: Thompson sampling, sequential choice, online recommendations, fatigue, regret upper bound.

Acknowledgements

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

1 Introduction

In applications such as email newsletters or app notifications, the platform's goal is to carefully tailor items so as to maximize revenue while maintaining user retention. Both these metrics depend not only on the intrinsic quality of the items themselves, but also on the way they are positioned when the users view them. When the user's precise behavior is not known a priori, the platform may have to learn and maximize revenue simultaneously. In many such platforms, users can be categorized into different types and the platform has the ability to interact with multiple users of the same type sequentially and independently. If the items are well aligned with the interests of the users, then the platform benefits from increased sales, its brand gets promoted and may also cause steady user growth. On the other hand, if the items are not interesting to the users, then it may induce fatigue (a state where their perceived value of the platform decreases) leading to user abandonment (for instance, canceled subscriptions or app uninstalls).

In this paper, we consider the following setting: the platform needs to learn a sequence of items (from a set of N items) by interacting with its users in rounds. In particular, it wants to maximize its expected utility when compared to the best sequence in hindsight. When a user is presented with a sequence of items, they view it from top-to-bottom and at each position they make decisions: whether satisfy with the current items and whether abandon the platform. One could attempt to model the above problem using the stochastic Multi-armed Bandit (MAB) formalism, where the decision maker selects one arm out of (say N) arms in each round, and receives feedback in the form of a reward sampled from a reward distribution. In our setting, the platform can choose both the length of the sequence as well as the order of the items, and this is essentially a combinatorial problem in each round. The recommended sequence of items should balance the penalty of user abandonment versus the upside of user choosing a high revenue item. The probability of a user choosing a high revenue item is not independent of other items in the recommended list. We assume that the aforementioned user behavior has a particular parametric form (see Section 2), whose parameters are not known a priori.

In this extended abstract, we design a fatigue-aware online recommendation solution, which we call the *Sequential Bandit Online Recommendation System* (SBORS). SBORS is generated from Thompson Sampling (TS) based algorithm with posterior approximation and correlated sampling to control exploration-vs-exploitation trade-off. We give the regret upper bound of SBORS (Section 3) is $C_1N^2\sqrt{NT\log TR}+C_2N\sqrt{T\log TR\cdot\log T}+C_3N/R$ (here C_1, C_2 and C_3 are constants, and R is a tunable algorithm parameter that captures the number of times that we sample the relevant random variables, see Section 3). We also propose a variant of our algorithm in the contextual setting where the user's context/profile affects the preference and abandonment parameters (Section 4). Additional details for theoretical analysis and experiments are provided in the full version [4].

2 Model

Our setting is similar to that of [2]. Consider a platform containing N different items indexed by i. let its corresponding revenue be r_i if selected. User's intrinsic preference for an item i is denoted by u_i. After viewing each item from a recommended list, the user has a probability p of abandoning the platform, and the occurrence of this event causes the platform to incur a penalty cost c. Note that $r_i, u_i, p, c \in [0, 1]$. We represent the sequence of items at time/round t as $\mathbf{S}^t = (S_1^t, S_2^t, ..., S_m^t)$, where S_i^t denotes the i^{th} item, and m represents the length of the sequence.

After the user at time t sees item i, s/he has three options based on behavior parameters $\mathbf{u}$ and p: (1) The user is satisfied with the item i (perhaps clicks the item's link and navigates to a target page), then no further items are presented to the user. In this situation, the platform earns revenue r_i. (2) The user is not satisfied with item i and decides to see the following item $i+1$ in the sequence of items (for instance, the next notification) if it exists. When the sequence runs out, the user exits the platform. In this situation, the platform will neither earn a reward nor pay a penalty cost. (3) The user has lost interest in the platform (presumably after viewing uninteresting items) and s/he decides to abandons the platform (for instance, by uninstalling the app). In this situation, the platform incurs a penalty c.

The behavior parameters $\mathbf{u}$ and p parameterize the following distributions. Consider a random variable W^t following a distribution F_W. W^t measures the t^{th} user's patience, capturing the number of unsatisfied items the user sees without abandoning the platform. In particular, F_W is a geometric distribution with parameter p. Let $q = 1 - p$. Then $q_k = q^{k-1}(1-q)$ denotes the probability that a user abandons the platform after receiving k^{th} unsatisfying item. Further, let $\widetilde{F}_W(k) = P(W > k) = 1 - P(W \leq k) = q^k$ denote the probability that a user does not abandon the platform after receiving the k^{th} unsatisfying item. The probability of each item i being selected is u_i, which is only determined by its content. The probability of each item i being selected when it belongs to the sequence of items $\mathbf{S}$ (dropping the superscript t for simplicity) is denoted as $p_i(\mathbf{S})$. $p_i(\mathbf{S})$ not only depends on the item's intrinsic value to the user, but also depends on its position and the other items shown before it. The probability of total abandonment is denoted as $p_a(\mathbf{S})$, and represents the sum of the probabilities that the platform is abandoned after receiving k unsatisfying items. In

summary,

$$p_i(\mathbf{S}) = \begin{cases} u_i & \text{if } i \in S_1, \\ \widetilde{F}_W(l-1)\prod_{k=1}^{l-1}(1-u_{I(k)})u_i & \text{if } i \in S_l, l \geq 2, \\ 0 & \text{if } i \notin \mathbf{S}. \end{cases}$$

And $p_a(\mathbf{S}) = \sum_{k=1}^{m} q_k \prod_{j=1}^{k}(1-u_{I(j)})$, where $I(k)$ means that in the sequence of items $\mathbf{S}$, the k^{th} items is i, i.e. $S_k = \{i\}$. We denote $U(\mathbf{S}, \mathbf{u}, q)$ as the total utility (payoff) that the platform receives from a given sequence of items $\mathbf{S}$. The goal is to find the optimal sequence of items that can optimize the expected utility $\mathbb{E}[U(\mathbf{S}, \mathbf{u}, q)] = \sum_{i \in \mathbf{S}} p_i(\mathbf{S}) r_i - c p_a(\mathbf{S})$:

$$\max_{\mathbf{S}} \ \mathbb{E}[U(\mathbf{S}, \mathbf{u}, q)] \quad \text{s.t.} \quad S_i \cap S_j = \emptyset, \forall i \neq j. \tag{1}$$

The constraint above specifies that all the items contained in the sequence are distinct. We denote the optimal sequence of items for a given $\mathbf{u}, q$ pair using $\mathbf{S}^* = \arg\max_{\mathbf{S}} \mathbb{E}[U(\mathbf{S}, \mathbf{u}, q)]$. If it is not unique, ties are broken arbitrarily.

3 SBORS: Sequential Bandit for Online Recommendation System & Regret Analysis

We first describe an algorithm that captures the TS approach. Unfortunately, a direct analysis of this version is difficult, so we modify it suitably to design our proposed algorithm SBORS. Due to length limitation, we will not discuss the precursor to SBORS here. For more information, please see full version [4].

Denote $c_i(t)$ as the total number of users selecting item i, and $f_i(t)$ as the total number of users observing item i without selection. Let $T_i(t) = c_i(t) + f_i(t)$. Denote $n_a(t)$ as the number of users who abandon the platform by time t, $n_e(t)$ as the number of times that users do not select an item and do not abandonment by time t. Let $N_q(t) = n_e(t) + n_a(t)$. Let $I(\cdot)$ denote the index function such that $I(k) = i$ if and only if $S_k = i$. As shown in [2] (Lemma 5), we can get unbiased estimates of the true parameters as follows:

Lemma 1 *Unbiased estimates:* $\hat{u}_i(t) = \frac{c_i(t)}{T_i(t)}$ *is an unbiased estimator for* u_i *and* $\hat{q}_i(t) = \frac{n_e(t)}{N_q(t)}$ *is an unbiased estimator for* q.

Motivated by [1], in algorithm SBORS we maintain a Gaussian posterior distribution for the selection parameter u_i and the abandonment distribution parameter q, which we update as we observe the user's feedback to our current recommended list. We also perform correlated sampling (which boosts variance boosting and allows for a finer exploration-exploitation trade-off). For a user arriving at time t, we calculate the current optimal sequence of items based on samples $\mathbf{u}'(t)$ and $q'(t)$. When the sequence of items is shown, we observe the user's feedback, then update the corresponding parameters of the relevant Beta distributions.

Posterior approximation: We approximate the posteriors for u_i, q by Gaussian distributions with approximately the same mean and variance as the original Beta distributions. In particular, let

$$\hat{u}_i(t) = \frac{c_i(t)}{c_i(t)+f_i(t)} = \frac{c_i(t)}{T_i(t)}, \hat{\sigma}_{u_i}(t) \quad = \sqrt{\frac{\alpha \hat{u}_i(t)(1-\hat{u}_i(t))}{T_i(t)+1}} + \sqrt{\frac{\beta}{T_i(t)}}, \tag{2}$$

$$\hat{q}(t) = \frac{n_e(t)}{n_e(t)+n_a(t)} = \frac{n_e(t)}{N_q(t)}, \hat{\sigma}_q(t) \quad = \sqrt{\frac{\alpha \hat{q}(t)(1-\hat{q}(t))}{N_q(t)+1}} + \sqrt{\frac{\beta}{N_q(t)}}, \tag{3}$$

where $\alpha > 0, \beta \geq 2$ are constants, be the means and standard deviations of the approximating Gaussians.

Controlling exploration via correlated sampling: Instead of sampling $\mathbf{u}'$ and q' independently, we correlate them by using a common standard Gaussian sample and transforming it. That is, in the beginning of a round t, we generate a sample from the standard Gaussian $\theta \sim N(0,1)$, and the posterior sample for item i is computed as $\hat{u}_i(t) + \theta \hat{\sigma}_{u_i}(t)$, while the posterior sample for abandonment is computed as $\hat{q}(t) + \theta \hat{\sigma}_q(t)$. This allows us to generate sample parameters for $i = 1, \cdots, N$ that are highly likely to be either simultaneously high or simultaneously low. As a consequence, the parameters corresponding to items in the ground truth $\mathbf{S}^*$, will also be simultaneously high/low. Because correlated sampling decreases the joint variance of the sample, we can counteract by generating multiple Gaussian samples. In particular, we generate R independent samples from the standard Gaussian, $\theta^{(j)} \sim N(0,1)$, $j \in [R]$. We take the maximum from j samples. Precisely, the parameters are generated as:

$$u_i'(t) = \max_{j=1,\cdots,R} \hat{u}_i(t) + \theta^{(j)} \hat{\sigma}_{u_i}(t)^2, \text{ and } \quad q'(t) = \max_{j=1,\cdots,R} \hat{q}(t) + \theta^{(j)} \hat{\sigma}_q(t)^2$$

Then we solve the optimization problem to get $\mathbf{S}^t = \arg\max_{\mathbf{S}} \mathbb{E}[U(\mathbf{S}^t, \mathbf{u}'(t), q'(t))]$.

Our main result is shown in Theorem 1. We omit the theoretical analysis for brevity. Please refer to [4] for more details.

Algorithm 1 SBORS algorithm

Initialization: Set $c_i(t) = f_i(t) = 1$ for all $i \in X$; $n_e(t) = n_a(t) = 1$; $t = 1$;
while $t \leq T$ **do**
Update $\hat{u}_i(t), \hat{q}(t), \hat{\sigma}_{u_i}(t), \hat{\sigma}_q(t)$ from (2) and (3);
(a) *Correlated sampling*:
for $j = 1, ..., R$ **do**
Get $\theta^{(j)} \sim N(0,1)$ and compute $u_i^{\prime(j)}(t)$,$q^{\prime(j)}(t)$
For each $i \leq N$, compute $u_i'(t) = \max_{j=1,\cdots,R} u_i^{\prime(j)}(t)$ and $q'(t) = \max_{j=1,\cdots,R} q^{\prime(j)}(t)$.
(b) *Sequence selection*:
Compute $\mathbf{S}^t = \arg\max_{\mathbf{S}} \mathbb{E}[U(\mathbf{S}, \mathbf{u}'(t), q'(t))]$; Observe feedback upon seeing the $k_t = |\mathbf{S}^t|$ items;
(c) *Posterior update*:
for $j = 1, \cdots, k_t$ **do**
Update

$$(c_{I(j)}(t), f_{I(j)}(t), n_e(t), n_a(t)) = \begin{cases} (c_{I(j)}(t)+1, f_{I(j)}(t), n_e(t), n_a(t)) & \text{if select and leave} \\ (c_{I(j)}(t), f_{I(j)}(t)+1, n_e(t)+1, n_a(t)) & \text{if not select and not abandon} \\ (c_{I(j)}(t), f_{I(j)}(t)+1, n_e(t), n_a(t)+1) & \text{if not select and abandon} \end{cases}$$

$c_i(t+1) = c_i(t)$, $f_i(t+1) = f_i(t)$ for all i; $\quad n_e(t+1) = n_e(t)$, $n_a(t+1) = n_a(t)$
$t = t + 1$;

Theorem 1 ***(Main Result)*** *Over T rounds, the regret of SBORS (Algorithm 1) is bounded as:*

$$Reg(T; \mathbf{u}, q) \leq C_1 N^2 \sqrt{NT \log TR} + C_2 N \sqrt{T \log TR \cdot \log T} + \frac{C_3 N}{R},$$

where C_1, C_2 and C_3 are constants and R is an algorithm parameter.

4 SBORS in the Contextual Setting

We now consider a more realistic setting where the diversity of users affects the offered sequence of items. Let each user's information be denoted as a d dimensional vector $\mathbf{x}_t$ at round t, which determines this user's preference parameter $\mathbf{u}(\mathbf{x}_t)$ and abandonment parameter $q(\mathbf{x}_t)$. User's feedback for each item is whether it is selected or not, and the feedback for abandonment is also binary. Let $Ru_i(t)$ and $Rq(t)$ represent the feedback after showing a sequence $\mathbf{S}^t$ in round t. Thus, $Ru_i(t) = c_i(t) - c_i(t-1)$, $Rq(t) = n_a(t) - n_a(t-1)$. Since $Ru_i(t)$ and $Rq(t)$ are binary, we assume that a logistic map $\mu(x) = \frac{1}{1+e^{-x}}$ relates the context and the parameters. That is,

$$u_i(\mathbf{x}_t) = \frac{1}{1 + e^{-\gamma_i^T \mathbf{x}_t}}, \text{ and } q(\mathbf{x}_t) = \frac{1}{1 + e^{-\delta^T \mathbf{x}_t}}, \tag{4}$$

where γ_i and δ are unknown parameters. We can estimate γ_i (for each $i \in [N]$) and δ by solving the following equations at each round [3]:

$$\sum_{t=1}^{T-1} \mathbb{1}(i \in \mathbf{S}^t) \cdot \Big(Ru_i(t) - \mu(\gamma_i^T \mathbf{x}_t) \Big) \mathbf{x}_t = 0, \text{ and } \quad \sum_{t=1}^{T-1} \Big(Rq(t) - \mu(\delta^T \mathbf{x}_t) \Big) \mathbf{x}_t \quad = 0. \tag{5}$$

This allows us to extend SBORS (Algorithm 1) in a natural way (see Algorithm 2). To initialize, we offer each of item once to an initial set of users. Then in each round t, we first update $\gamma_i(t)$ and $\delta(t)$ based on (5). Next we compute $\widetilde{u}_i(t, \mathbf{x})$, $\widetilde{q}(t, \mathbf{x})$ and their corresponding $\widetilde{\sigma}$s, given context $\mathbf{x}$ using (4). Next, we sample $u_i'(t, \mathbf{x})$ and $q'(t, \mathbf{x})$ from Gaussians as before and solve $\max_{\mathbf{S}} \mathbb{E}[U(\mathbf{S}, \mathbf{u}_t(\mathbf{x}_t), q_t(\mathbf{x}_t))]$.

5 Empirical Experiments & Conclusion

In this section, we only demonstrate the robustness of Algorithm 1 by comparing how the regret changes with respect to different values of $\mathbf{u}$ and other relevant parameters. For more scenarios and discussion, please refer to [4].

Setting: $N = 30$, reward r_i is uniformly distributed between $[0, 1]$, abandonment distribution probability $p = 0.1$ and the cost of abandonment $c = 0.5$. Additionally, we generate $\mathbf{u}$ form [0,0.1], and discuss the influence of sampling parameter R, and fixed constants α, β on the regret separately.

Algorithm 2 Contextual SBORS algorithm

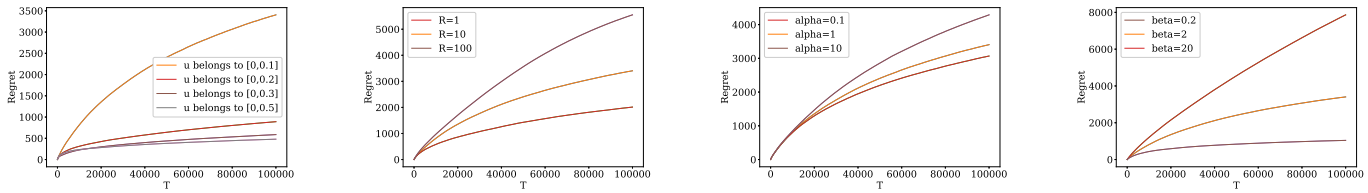
Initialization: Set $c_i(t) = f_i(t) = 1$ for all $i \in [N]$; $n_e(t) = n_a(t) = 1$; $t = 1$;
Offer each item to users $1, \cdots, N$ and collect feedback. Update $t = N$;
while $t \leq T$ **do**
(a) *Posterior sampling*:
Update $\gamma_i(t)$ and $\delta(t)$ by quasi-MLE according to (5); Observe user's contextual information $\mathbf{x}_t$
Get $\widetilde{u}_i(t, \mathbf{x}_t)$ and $\widetilde{q}(t, \mathbf{x}_t)$ according to (4), and
$\widetilde{\sigma}_{u_i}(t, \mathbf{x}_t) = \sqrt{\frac{\alpha \widetilde{u}_i(t,\mathbf{x}_t)(1-\widetilde{u}_i(t,\mathbf{x}_t))}{T_i(t)+1}} + \sqrt{\frac{\beta}{T_i(t)}}$, $\widetilde{\sigma}_q(t, \mathbf{x}_t) = \sqrt{\frac{\alpha \widetilde{q}(t,\mathbf{x}_t)(1-\widetilde{q}(t,\mathbf{x}_t))}{N_q(t)+1}} + \sqrt{\frac{\beta}{N_q(t)}}$,
where $T_i(t) = c_i(t) + f_i(t)$, $N_q(t) = n_e(t) + n_a(t)$, and α and β are the same as Algorithm 1;
For each item $i \in [N]$, sample $u_i'(t, \mathbf{x}_t)$ and $q'(t, \mathbf{x}_t)$ from the corresponding Gaussian distribution:
$u_i'(t, \mathbf{x}_t) \sim N(\widetilde{u}_i(t, \mathbf{x}_t), \widetilde{\sigma}_{u_i}(t, \mathbf{x}_t))$; $q'(t, \mathbf{x}_t) \sim N(\widetilde{q}(t, \mathbf{x}_t), \widetilde{\sigma}_q(t, \mathbf{x}_t))$
(b) *Sequence selection*:
Compute $\mathbf{S}^t = \arg\max_{\mathbf{S}} \mathbb{E}[U(\mathbf{S}, \mathbf{u}'(t, \mathbf{x}_t), q'(t, \mathbf{x}_t))]$; Offer $\mathbf{S}^t$ and observe feedback.
(c) *Posterior update*: Same as step (c) of Algo. 1.

Influence of u: We present four scenarios, when the preference parameter $\mathbf{u}$ is uniformly generated from $[0, 0.1]$, $[0, 0.2]$, $[0, 0.3]$, $[0, 0.5]$, element-wise. Fig. 1(a) shows that the regrets eventually tend to stop growing steeply. The more $\mathbf{u}$ is spread out, the lower the regret is.

Influence of R: We set $\alpha = 1$, $\beta = 2$ and vary R. Fig. 1(b) shows that lower R values reduce the regret. One extreme case is $R = 1$, which essentially removes variance boosting and still performs well empirically.

Influence of α: We set $R = 10$, $\beta = 2$ and change α. Fig. 1(c) shows that lower α values reduce the regret.

Influence of β: We set $R = 10$, $\alpha = 1$ and change β. Fig. 1(d) shows that lower β reduce regret. For analysis, we needed $\beta \geq 2$, but we observe that choosing $\beta < 2$ can still lead to better regret hinting at a potential slack in our analysis.



(a) u_i is uniformed generated from 0 to $0.1, 0.2, 0.3, 0.5$, resp.

(b) R is 1, 10, 100, resp.

(c) α is 0.1, 1, 10, respectively.

(d) β is 0.2, 2, 20, respectively.

Figure 1: Plots for SBORS with different $\mathbf{u}$, R, α, and β.

In this extended abstract, we present a new Thompson Sampling based algorithm for making recommendations where users experience fatigue. We use techniques such as posterior approximation using Gaussians, correlate sampling and variance boosting to control the exploration-exploitation trade-off and derive rigorous regret upper bounds. Our bounds depend polynomially on the number of items and sub-linearly on the time horizon ($C_1 N^2 \sqrt{NT \log TR} + C_2 N \sqrt{T \log TR \cdot \log T} + C_3 N/R$). Our algorithm is easily extendable to the contextual setting.

References

[1] Shipra Agrawal, Vashist Avadhanula, Vineet Goyal, and Assaf Zeevi. Thompson sampling for the mnl-bandit. *arXiv preprint arXiv:1706.00977*, 2017.

[2] Junyu Cao and Wei Sun. Dynamic learning of sequential choice bandit problem under marketing fatigue. *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.

[3] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits: The generalized linear case. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 586–594. Curran Associates, Inc., 2010.

[4] Yunjuan Wang and Theja Tulabandhula. Thompson sampling for a fatigue-aware online recommendation system. *ArXiv Preprint*, abs/1901.07734, 2019.

Privacy-preserving Q-Learning with Functional Noise in Continuous State Spaces

Baoxiang Wang **Nidhi Hegde**

Borealis AI, Edmonton, Canada

{ baoxiang.wang, nidhi.hegde } @ borealisai.com

Abstract

We consider privacy-preserving algorithms for reinforcement learning with continuous state spaces. The aim is to release the value function which does not distinguish two neighboring reward functions $r(\cdot)$ and $r'(\cdot)$. Existing studies that guarantee differential privacy are not extendable to infinity state spaces, since the noise level to ensure privacy will scale accordingly. We use functional noise, which protects the privacy for the entire value function approximator, without regard to the number of states queried to the function. With analyses on the RKHS of the functional, the uniform bound such samples noise and the composition of iteratively adding the noise, we show the rigorous privacy guarantee. Under the discrete space setting, we gain insight by analyzing the algorithm's utility guarantee. Experiments corroborate our theoretical findings.

Our code is available at `https://github.com/wangbx66/differentially-private-q-learning`. For all the technical details the full paper is at `https://arxiv.org/abs/1901.10634`.

Keywords: differential privacy, RKHS, reinforcement learning, q-learning

1 Introduction

Increasing interest in reinforcement learning (RL) and deep reinforcement learning has led to recent advances in a wide range of algorithms [SB18]. While a large part of the advancement has been conducted on the space of games, the applicability of RL extends to other practical cases such as recommendations and search engines [BHM12]. With the popularity of the RL algorithms raised the concerns about their privacy. Namely, the released policy function and value function are trained based on the reward signal and other inputs, which are commonly relying on sensitive data. For example, an RL recommendation system may use the reward signals simulated by users' historical records. This historical information can thus be inferred by recursively querying the released functions.

RL methods learn by carrying out actions, receiving rewards observed for that action in a given state, and transitioning to the next states. Observation of the learned policy and value can reveal the following sensitive information: *the reward function* is a succinct description of the task and the preference; *the visited states* may carry important contextual information on the users, such as age, gender occupation, and etc.; *the transition function* includes the dynamic of the system and the impact of actions on the dynamic. Among those, the reward function is the most vulnerable and valuable component. The former is by the fact that the reward involves the least amount of randomness. The later is due to the rich information contained in the reward function that directly describes the users. In this paper the aim is to protect the reward function.

In order to achieve this under continuous space settings, we investigate the Gaussian process mechanism, proposed by [HRW13]. The mechanism provides functional noise-adding to the value function approximation hence the function can be evaluated at arbitrary many states while preserving the privacy. The mechanism requires a reproducing kernel Hilbert space (RKHS) and our choice of RKHS ensures that neural networks are included in the functional space hence can be used as the approximation. With the use of the mechanism, we can modify the deep Q-learning algorithm [MKS+15, WD92] so that the value function is protected after each update. The privacy guarantee follows a series of techniques elaborated.

Utility analysis of our algorithm is given under the tractable discrete state space case to gain some insight.

We provide an implementation of our algorithms and our experiments corroborate the theoretical findings.

Related Works. There is a significant line of research that discusses privacy-preserving approaches on online learning and bandits [SB18, Sze10]. The algorithms protect the neighboring rewards sequences from being distinguished, which is close to our definition of neighboring reward functions. The works [TD17, MT15, TS13] share the similar motivation as our work, but they do not scale to the continuous space due to the $\sqrt{N}$ or $\sqrt{N \log N}$ factor involved where N is the number of arms. Similarly motivated are the the online learning settings [GUK17, ALMT17, AS17], where analyses are based on optimizing a fixed objective and thus will not apply to our setting.

More closely related are studies on contextual bandits [SS19, SS18, LWZC16], where a contextual vector is analogous to the states in reinforcement learning. Equivalently, policy evaluation [BGP16] considers a similar setting where the value function is learned for a one-step MDP. The major challenge to extend these work is that reinforcement learning, which is an iterative process of policy evaluation and policy improvement, requires access to the reward sequence. Our proposed approach will be for both the evaluation and the improvement, while also extending it to non-linear approximations.

A general approach that can be applied to continuous spaces is the differential privacy deep learning framework [ACG+16]. The method perturbs the gradient estimator in the updates of neural networks to preserve privacy. In our problem, applying this method will require a large noise level which makes it unrealistic to keep a competitive performance. In fact, the framework relies on the setting where at most one data point can be different in the neighboring inputs. It therefore benefits from a $1/B$ factor on the noise level where B is the batch size. This no longer holds as in reinforcement learning all reward signals can be different for neighboring reward functions, causing the noise to scale B times.

2 Preliminaries

2.1 Notations on Markov Decision Process and Reinforcement Learning

An MDP includes the state space $\mathcal{S}$, the action space $\mathcal{A} = \{1, \ldots, m\}$, the transition kernel $\mathcal{T}$, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the discount factor $\gamma \in [0, 1)$. We investigate bounded and continuous state space $\mathcal{S} \subseteq \mathbb{R}$ and without loss of generality assume that $\mathcal{S} = [0, 1]$. Denote the corresponding action-state value function as $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t \geq 0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a, \pi]$. When the context is clear, we omit π and write $Q(s, a)$ instead.

2.2 Differential Privacy

Differential privacy [DKM+06, DMNS06] ensures that data analysis should not differ at the aggregate level whether any given user is present in the input or not. This latter condition on the presence of any user is formalized through the notion of neighboring inputs. In this paper, r and r' are considered neighboring inputs if $\|r - r'\|_\infty \leq 1$.

Definition 1. *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{U}$ satisfies (ϵ, δ)-differential privacy if for any two neighboring inputs d and d' and for any subset of outputs $\mathcal{Z} \subseteq \mathcal{U}$ it holds that*

$$\mathbb{P}(\mathcal{M}(d) \in \mathcal{Z}) \leq \exp(\epsilon)\mathbb{P}(\mathcal{M}(d') \in \mathcal{Z}) + \delta.$$

An important parameter of a mechanism is the (global) sensitivity of the output.

Definition 2. *For all $d, d' \in \mathcal{D}$ neighboring inputs, the sensitivity of a mechanism $\mathcal{M}$ is defined as*

$$\Delta_{\mathcal{M}} = \max_{d,d' \in \mathcal{D}} \|\mathcal{M}(d) - \mathcal{M}(d')\|, \tag{1}$$

where $\|\cdot\|$ is some norm defined on $\mathcal{U}$.

Vector-output mechanisms. For converting vector-valued functions into a (ϵ, δ)-DP mechanism, one of the standard approaches is the Gaussian mechanism. This mechanism adds $\mathcal{N}(0, \sigma^2\mathbb{I})$ to the output $\mathcal{M}(d)$. In this case $\mathcal{U} = \mathbb{R}^n$ and $\|\cdot\|$ in (1) is the ℓ^2-norm $\|\cdot\|_2$ of the Euclidean space.

Proposition 3 (Vector-output Gaussian mechanism [DR14])**.** *If $0 < \epsilon < 1$ and $\sigma \geq \sqrt{2\ln(1.25/\delta)}\Delta_{\mathcal{M}}/\epsilon$, then $\mathcal{M}(d) + y$ is (ϵ,δ)-differentially private, where y is drawn from $\mathcal{N}(0, \sigma^2 I)$.*

Function-output mechanisms. In this setting the output of the function is a function, which means the mechanism is a functional. We consider the case where $\mathcal{U}$ is an RKHS and $\|\cdot\|$ in (1) is the RKHS norm $\|\cdot\|_{\mathcal{H}}$. Hall et al. [HRW13] have shown that adding a Gaussian process noise $\mathcal{G}(0, \sigma^2 K)$ to the output $\mathcal{M}(d)$ is differentially private, when K is the RKHS kernel of $\mathcal{U}$.

Proposition 4 (Function-output Gaussian process mechanism [HRW13])**.** *If $0 < \epsilon < 1$ and $\sigma \geq \sqrt{2\ln(1.25/\delta)}\Delta_{\mathcal{M}}/\epsilon$, then $\mathcal{M}(d) + g$ is (ϵ,δ)-differentially private, where g is drawn from $\mathcal{G}(0, \sigma^2 K)$ and $\mathcal{U}$ is an RKHS with kernel function K.*

3 Differentially Private Q-Learning

3.1 Our Algorithm

Algorithm 1 Differentially Private Q-Learning with Functional Noise

1: **Input:** the environment and the reward function $r(\cdot)$
2: **Parameters:** target privacy (ϵ, δ), time horizon T, batch size B, action space size m, learning rate α, reset factor J
3: **Output:** trained value function $Q_\theta(s, a)$
4: **for** j **in** $[T/B]$ **do**
5: $\quad \hat{g}_k[B][2] \leftarrow \{\}$ if $j \equiv 0 \mod T/JB$;
6: $\quad$ **for** b **in** $[B]$ **do**
7: $\quad\quad t \leftarrow jT/B + b$;
8: $\quad\quad$ Execute $a_t = \arg\max_a Q_\theta(s_t, a) + \hat{g}_a(s_t)$;
9: $\quad\quad$ Receive r_t and s_{t+1}, $s \leftarrow s_{t+1}$;
10: $\quad\quad$ **for** $a \in [m]$ **do**
11: $\quad\quad\quad$ Insert s to $\hat{g}_a[:][1]$ such that the list is increasing;
12: $\quad\quad\quad$ Sample $z_{at} \sim \mathcal{N}(\mu_{at}, \sigma d_{at}))$, according to Eq. (2), Appendix A;
13: $\quad\quad\quad$ Update the list $\hat{g}_a(s) \leftarrow z_{at}$;
14: $\quad\quad$ **end for**
15: $\quad\quad y_t \leftarrow r_t + \gamma \max_a Q_\theta(s_{t+1}, a) + \hat{g}_a(s_{t+1})$;
16: $\quad\quad l_t \leftarrow \frac{1}{2}(Q_\theta(s_t, a_t) + \hat{g}_a(s_t) - y_t)^2$;
17: $\quad$ **end for**
18: $\quad$ Run one step SGD $\theta \leftarrow \theta + \alpha\frac{1}{B}\nabla_\theta \sum_{t=jB}^{(j+1)B} l_t$;
19: **end for**

We present our algorithm for privacy-preserving Q-learning under the setting of continuous state space. We perturb the updated value function at each iteration by adding a Gaussian process noise. This is described by line 18 and 19 of the algorithm, where $\hat{g}$ is the noise. Line 13-17 describes the necessary steps for $\hat{g}$ to simulate the Gaussian process. Line 6-8 re-sample a Gaussian process sample path for every J iterations. Other steps are similar to [MKS+15].

Insights into the algorithm design. We require two reward functions r and r' to not be distinguished by observing the learned functions. Since the reward signal $r(s, a)$ can appear at any s. Therefore, we need a stronger mechanism that covers the entire state space, which leads to our utilization of the tools provided by Hall et al. [HRW13].

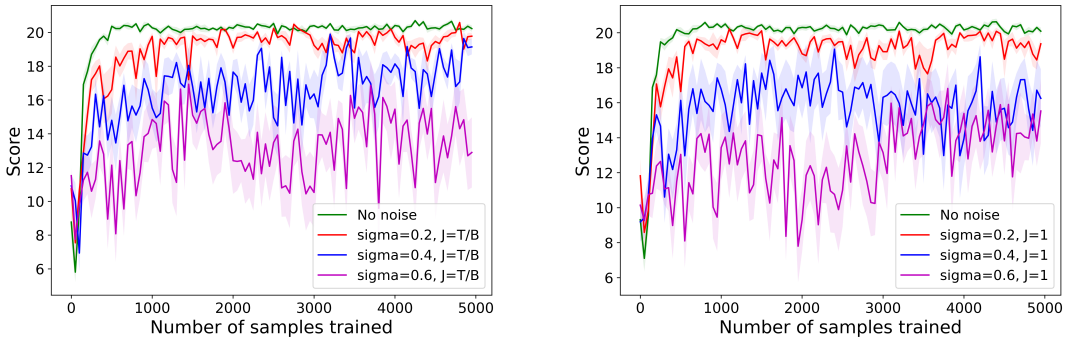


Figure 1: Empirical results on differentially private q-learning. The y-axis denotes the score. The x-axis is the number of samples the agent has trained on. Averaged over 10 random seeds.

3.2 Privacy, Efficiency, and Utility of the Algorithm

Privacy analysis. There are three main components in the privacy analysis. Analysis of the RKHS, composition of the mechanism, and the uniform bound of the noise. With these components our privacy guarantee is shown in the below Theorem 5.

Theorem 5. *The Q-learning algorithm in Algorithm 1 is* $(\epsilon, \delta + J\exp(-(2k - 8.68\sqrt{\beta}\sigma)^2/2))$*-DP, provided that* $2k > 8.68\sqrt{\beta}\sigma$*, and*

$$\sigma \geq \sqrt{2(T/B)\ln(e + \epsilon/\delta)}C(\alpha, k, L, B)/\epsilon,$$

where $C(k, L, B) = ((4\alpha(k+1)/B)^2 + 4\alpha(k+1)/B)L^2$*,* L *is the Lipschitz constant of the value function approximation,* B *is the batch size and* T *is the number of iterations, and* α *is the learning rate.*

Time complexity. We show that the noise adding in our algorithm is efficient. In fact, the most complex step introduce by the noise-adding is the insertion in line 14, which takes logarithmic time.

Proposition 6. *The noised value function in Algorithm 1 can respond to* N_q *rounds of queries in* $\mathcal{O}(\log(N_q))$ *time.*

Utility analysis. To the best of our knowledge, there is no study to rigorously analyze the utility of deep reinforcement learning. However, we gain insight by analyzing the algorithm's learning error in the discrete state space setting.

Proposition 7. *Let* v' *and* v^* *be the value function learned by our algorithm and the optimal value function, respectively. In the case* $J = 1$*,* $|S| = n < \infty$*, and* $\gamma < 1$*, the utility loss of the algorithm satisfies* $\mathbb{E}[\frac{1}{n}\|v' - v^*\|_1] \leq \frac{2\sqrt{2}\sigma}{\sqrt{n\pi}(1-\gamma)}$.

3.3 Discussion

Extending to other RL algorithms. Our algorithm can be extended towards the actor-critic method and its variants [MBM+16, YWT18]. Any post-processing of the private Q function will not break the privacy guarantee, including experience replay and ϵ-greedy policies [MKS+15]. In the case where the reward is directly accessed in the policy gradient estimation [SML+15, LW18, PFW+18] one should add noise to the policy function as well.

Extending to high-dimensional tasks. Our approach can also be extended to high-dimension spaces by choosing a high dimensional RKHS. For example, the kernel function $\exp(-\beta\|x - y\|_1)$ where $\|\cdot\|_1$ is now the Manhattan distance. It is also possible to use other RKHS for the Gaussian process noise, such as the space of band-limited functions.

4 Experiments

We test Algorithm 1 using our designed MDP settings and plot the learning curve with a variety of noise levels and J values in Figure 1. Intuitively, with the increase of the noise level, the algorithm takes more samples to achieve the performance of the non-noisy version. However, we observe that with the noise being reset every round ($J = T/B$), the algorithm is likely to converge with limited sub-optimality.

References

[ACG+16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[ALMT17] Jacob Abernethy, Chansoo Lee, Audra McMillan, and Ambuj Tewari. Online linear optimization through the differential privacy lens. *arXiv preprint arXiv:1711.10019*, 2017.

[AS17] Naman Agarwal and Karan Singh. The price of differential privacy for online learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 32–40. JMLR.org, 2017.

[BGP16] Borja Balle, Maziar Gomrokchi, and Doina Precup. Differentially private policy evaluation. In *Proceedings of the 33th International Conference on Machine Learning*, pages 2130–2138, 2016.

[BHM12] Siddhartha Banerjee, Nidhi Hegde, and Laurent Massoulié. The price of privacy in untrusted recommendation engines. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 920–927. IEEE, 2012.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[GUK17] Pratik Gajane, Tanguy Urvoy, and Emilie Kaufmann. Corrupt bandits for preserving local privacy. *arXiv preprint arXiv:1708.05033*, 2017.

[HRW13] Rob Hall, Alessandro Rinaldo, and Larry Wasserman. Differential privacy for functions and functional data. *Journal of Machine Learning Research*, 14(Feb):703–727, 2013.

[LW18] Jiajin Li and Baoxiang Wang. Policy optimization with second-order advantage information. *arXiv preprint arXiv:1805.03586*, 2018.

[LWZC16] Shuai Li, Baoxiang Wang, Shengyu Zhang, and Wei Chen. Contextual combinatorial cascading bandits. In *ICML*, volume 16, pages 1245–1253, 2016.

[MBM+16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33th International Conference on Machine Learning*, pages 1928–1937, 2016.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[MT15] Nikita Mishra and Abhradeep Thakurta. (nearly) optimal differentially private stochastic multi-arm bandits. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 592–601. AUAI Press, 2015.

[PFW+18] Yangchen Pan, Amir-massoud Farahmand, Martha White, Saleh Nabi, Piyush Grover, and Daniel Nikovski. Reinforcement learning with function-valued action spaces for partial differential equation control. *arXiv preprint arXiv:1806.06931*, 2018.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SML+15] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[SS18] Roshan Shariff and Or Sheffet. Differentially private contextual linear bandits. In *Advances in Neural Information Processing Systems*, pages 4301–4311, 2018.

[SS19] Touqir Sajed and Or Sheffet. An optimal private stochastic-mab algorithm based on an optimal private stopping rule. In *Proceedings of the 36th International Conference on Machine Learning*. JMLR.org, 2019.

[Sze10] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[TD17] Aristide Charles Yedia Tossou and Christos Dimitrakakis. Achieving privacy in the adversarial multi-armed bandit. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[TS13] Abhradeep Guha Thakurta and Adam Smith. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In *Advances in Neural Information Processing Systems*, pages 2733–2741, 2013.

[WD92] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[YWT18] Kenny Young, Baoxiang Wang, and Matthew E Taylor. Metatrace actor-critic: Online step-size tuning by meta-gradient descent for reinforcement learning control. *arXiv preprint arXiv:1805.04514*, 2018.

PAC-Bayesian Analysis of Counterfactual Risk in Stochastic Contextual Bandits

Junhao Wang
McGill University / Mila
junhao.wang@mail.mcgill.ca

Bogdan Mazoure
McGill University / Mila
bogdan.mazoure@mail.mcgill.ca

Gavin McCracken
McGill University / Mila
gavin.mccracken@mail.mcgill.ca

David Venuto
McGill University / Mila
david.venuto@mail.mcgill.ca

Audrey Durand
McGill University / Mila
audrey.durand@mcgill.ca

Abstract

This work tackles the off-policy evaluation problem within the contextual bandit setting, where only the action and reward recommended by the logging policy were recorded and thus available at evaluation. This kind of situation is encountered in applications where one wants to compute the optimal policy using data previously collected in an offline manner. Previous work have extended the PAC-Bayesian analysis to this setting, providing bounds on the clipped importance sampling risk estimator using a recent regularization technique known as *counterfactual risk minimization*. The contribution of this work is to tighten this existing result through the application of various PAC-Bayesian concentration inequalities: Kullback-Leibler divergence, Bernstein, and Azuma-Hœffding. This yields bounds on the empirical risk estimator that either converge at a faster rate given the amount of prior data, or that are more robust to the clipping factor.

Keywords: contextual bandits, PAC-Bayes

Acknowledgements

The authors would like to thank Vincent Luczkow and Nadeem Ward for insightful discussions.

1 Introduction

In many applications of interactive learning, one wants to leverage previously collected data in an offline manner in order to compute the optimal policy at a later stage. For instance, this is the case of recommender systems, where data may have been gathered previously under some recommendation policy (e.g. best top results) and one wants to use this data to evaluate an *alternative* recommendation policy. This is known as *off-policy* or *offline evaluation*. In this work, we will tackle the off-policy evaluation problem within the contextual bandit setting, where only the action and reward recommended by the logging policy were recorded and thus available at evaluation (Li et al., 2011).

The PAC-Bayesian analysis (Shawe-Taylor and Williamson, 1997; Shawe-Taylor et al., 1998; McAllester, 1999) has been dominantly focusing on studying supervised setting of statistical learning, where data is assumed to be independently and identically distributed (i.i.d), within the PAC (Probably Approximately Correct) learning framework (Valiant, 1984). Such analysis highlights the trade-off between the complexity of individual models from the hypothesis space and their empirical performance, with high probability guarantees on their expected performance. Seldin et al. (2011, 2012) have extended the framework to non-i.i.d setting such as bandits and reinforcement learning. Additionally, London and Sandler (2018) applied PAC-Bayesian analysis on a recent regularization technique known as *counterfactual risk minimization* (Swaminathan and Joachims, 2015) for off-policy evaluation on stochastic contextual bandits. The contribution of this work is to tighten this existing result through the application of various PAC-Bayesian concentration inequalities.

2 Contextual bandits

The stochastic contextual bandit (Langford and Zhang, 2008) is described by an arbitrary context space $\mathcal{X}$, an action space $\mathcal{K} = \{1, \dots, K\}$, and a distribution $\mathcal{D}$ over tuples (x, ρ), with $x \in \mathcal{X}$ and $\rho : \mathcal{X} \times \mathcal{K} \mapsto \mathcal{Y}$. Without loss of generality, we will assume in the following that $\mathcal{Y} = [0, 1]$. The problem can then be formulated as an episodic game where on each episode $t \in \mathbb{N}_{>0}$:

1. a context and reward function $(x_t, \rho_t \sim \mathcal{D})$ are generated from the environment;
2. the learner observes the context $x_t \in \mathcal{X}$ but does not observe the function ρ_t;
3. the learner selects an action $k_t \in \mathcal{K}$;
4. the learner observes the reward $y_t = \rho_t(x_t, k_t)$
5. the learner updates its knowledge based on this experience.

Assuming that $\mathcal{K}$ is finite, the goal of the learner is to learn a policy $\pi : \mathcal{X} \mapsto \Delta(\mathcal{K})$ for choosing actions over the probability simplex according to the context such that to maximize the expected reward $G(\pi) = \mathbb{E}_{(x_t,\rho_t)\sim\mathcal{D}}\mathbb{E}_{k_t\sim\pi(x_t)}\rho_t(x_t, k_t)$. This is equivalent to minimizing the *counterfactual risk*: $R(\pi) = 1 - G(\pi)$.

3 Off-policy evaluation

The task of off-policy policy evaluation consists in estimating either the true expected reward $G(\pi)$ or the true counterfactual risk $R(\pi)$ of an arbitrary policy π based on a n-length history $\mathcal{F}_n^{\pi_0} = \{(x_1, k_1, y_1), \dots, (x_n, k_n, y_n)\}$ generated by some policy π_0. This is often referred to as *learning from logged bandit feedback* (Li et al., 2011, 2012; Mary et al., 2014). For simplicity, we will only focus on analyzing the counterfactual risk R due to its similarity to expected reward G.

Assumption 1 (Time-invariance of π_0)**.** We assume that the logging policy π_0 is stationary, such that for every timestep $i \leqslant n$, the corresponding action $k_i \in \mathcal{F}_n^{\pi_0}$ has been sampled from the initial policy π_0.

A challenge in off-policy evaluation is to derive consistent estimators $\hat{R}(\pi, \mathcal{F}_n^{\pi_0})$ for some policy $\pi \neq \pi_0$, with low bias with regard to true counterfactual $R(\pi)$ and low variance with regard to the any history $\mathcal{F}_n^{\pi}$ for any policy π. Main methods for creating such estimators are direct modelling (Hassanpour and Greiner, 2018), importance sampling (Kearns et al., 2000; Precup, 2000), and doubly robust (Dudík et al., 2011), which combines the two previous to produce an estimator with lower variance and bias. This work focuses on the importance sampling (IS) counterfactual risk estimator

$$\hat{R}_{\mathrm{IS}}(\pi, \mathcal{F}_n^{\pi_0}) = 1 - \mathbb{E}_{(x_i,\rho_i)\sim\mathcal{D}}\left[\mathbb{E}_{k_i\sim\pi(x_i)}\left[\frac{\rho_i(x_i, k_i)}{\pi_0(x_i, k_i)}\right]\right] \approx 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{\pi(x_i, k_i)}{\pi_0(x_i, k_i)}\rho_i(x_i, k_i), \tag{1}$$

More specifically, we consider one important variant, that is the clipped importance sampling with counterfactual risk minimization objective (Hassanpour and Greiner, 2018).

Clipped importance sampling The essence of this approach is to set a lower bound on the propensity score $\pi_0(x_i, k_i)$, resulting in a clipped importance sampling risk estimator (CIS):

$$\hat{R}_{\mathrm{CIS}}(\pi, \mathcal{F}_n^{\pi_0}) = 1 - \frac{1}{n}\sum_{i=1}^{n} \underbrace{\frac{\pi(x_i, k_i)}{\max\{\pi_0(x_i, k_i), p_{\min}\}}\rho_i(x_i, k_i)}_{y_i^{\mathrm{CIS}}}. \tag{2}$$

Clipping by $p_{\min}$ trades off variance for bias in the estimator. Empirical variance regularizer for the clipped importance sampling weighted reward y_i^{CIS} can be applied to lead to faster shrinking of the difference between true counterfactual risk and empirical counterfactual risk, in comparison to without such regularizer. The variance penalty term, known as counterfactual risk minimization (CRM), is based on the following generalization error bound.

Theorem 3.1 (Counterfactual Risk Minimization (Swaminathan and Joachims, 2015))**.** *Let* Π *denote the space of policies.*

$$R(\pi) \leqslant \hat{R}_{\mathrm{CIS}}(\pi, \mathcal{F}_n^{\pi_0}) + O\left(\sqrt{\frac{\hat{\mathbb{V}}[\hat{R}_{\mathrm{CIS}}(\pi, \mathcal{F}_n^{\pi_0})] + \mathbb{C}(\Pi)}{n}} + \frac{\mathbb{C}(\Pi)}{n}\right), \tag{3}$$

where $\mathbb{C}(\Pi) \propto \mathcal{N}_\infty(\varepsilon, \Pi)$ *measures the cardinality of the minimal* $\varepsilon-$*covering of* Π*, and* $\hat{\mathbb{V}}[\hat{R}_{\mathrm{CIS}}(\pi, \mathcal{F}_n^{\pi_0})]$ *is the unbiased sample variance of the CIS risk estimator.*

4 PAC-Bayesian Counterfactual Risk Minimization

London and Sandler (2018) provide a Bayesian perspective of CRM by applying PAC-Bayesian analysis (McAllester, 1999) on contextual bandits, in a manner similar to Seldin et al. (2011). They achieve the following CRM bound, which depends on prior distribution $\mathbb{P}$ and posterior distribution $\mathbb{Q}$ over known deterministic hypothesis space $\mathcal{H} : \mathcal{X} \to \mathcal{K}$ such that $\pi_{\mathbb{Q}}(x, k) = \mathbb{E}_{h\sim\mathbb{Q}}[\mathbb{I}\{h(x) = k\}]$ corresponds to the posterior probability that a random hypothesis h maps an action k to context x. From the PAC-Bayesian perspective, the learner, which can be seen as a Gibbs classifier, samples h from $\mathbb{Q}(\mathcal{H})$ and selects action $k = h(x)$. In the off-policy evaluation setting considered in this work, the logging policy π_0 induced by $\mathbb{P}$ generated the history $\mathcal{F}_n^{\pi_0}$, and the goal consists in estimating $R(\pi)$ using $\hat{R}_{\mathrm{CIS}}(\pi, \mathcal{F}_n^{\pi_0})$.

Theorem 4.1 (PAC-Bayesian Counterfactual Risk Minimization (London and Sandler, 2018))**.** *Let* $\mathcal{H} \subseteq \{h : \mathcal{X} \to \mathcal{K}\}$ *denote a hypothesis space mapping contexts to actions and let* $\mathrm{KL}(\mathbb{Q}||\mathbb{P})$ *denote the Kullback-Leibler divergence between (absolutely continuous) probability measures* $\mathbb{Q}, \mathbb{P}$ *over the set* $\mathcal{H}$*. In particular, let* $\mathbb{P}$ *and* $\mathbb{Q}$ *be the prior and posterior distributions over* $\mathcal{H}$*, respectively. For any* $n \geqslant 1, \delta \in (0, 1), p_{\min} \in (0, 1)$ *, with probability at least* $1 - \delta$ *over* $\mathcal{F}_n^{\mathbb{P}}$*, which is history generated by* π_0 *induced by* $\mathbb{P}$*, the following holds simultaneously for all* $\mathbb{Q}$ *and its corresponding induced* $\pi_{\mathbb{Q}}$*:*

$$R(\pi_{\mathbb{Q}}) \leqslant \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}) + \sqrt{\frac{2(\frac{1}{p_{\min}} - 1 + \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}))(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n}{\delta})}{p_{\min}(n-1)}} + \frac{2(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n}{\delta})}{p_{\min}(n-1)} \tag{4}$$

Note 1. When $\hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}) = 1 - \frac{1}{p_{\min}}$, the generalization bound yields $O(\frac{1}{n})$ converging rate. In particular, minimizing $\hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\pi_{\mathbb{P}}})$ and keeping policy $\pi_{\mathbb{Q}}$ close to logging policy $\pi_{\mathbb{P}}$ through the KL minimizes $R(\pi_{\mathbb{Q}})$.

This result is obtained using the PAC-Bayesian-Hœffding inequality (McAllester, 2003). The following proposed result tightens the bound using various concentration inequalities.

4.1 Proposed result

By applying PAC-Bayes-KL inequality (Seeger, 2002), PAC-Bayes-Berstein inequality (Tolstikhin and Seldin, 2013), and PAC-Bayes-Azuma-Hœffding inequality (Seldin et al., 2012) (Theorems A.1,A.3,and A.2) on PAC-Bayesian CRM (Theorem 4.1), we obtain the following result.

Theorem 4.2 (PAC-Bayesian Counterfactual Risk Minimization Extensions)**.** *Let* $\mathcal{H} \subseteq \{h : \mathcal{X} \to \mathcal{K}\}$ *denote a hypothesis space mapping contexts to actions. For any* $n \geqslant 1, \delta \in (0, 1), p_{\min} \in (0, 1)$ *and fixed prior,* $\mathbb{P}$ *on* $\mathcal{H}$ *and its corresponding induced*

policy π_0, with probability at least $1-\delta$ over $\mathcal{F}_n^{\mathbb{P}}$, the following bounds holds simultaneously for all $\mathbb{Q}$ with induced policy $\pi_{\mathbb{Q}}$:

$$\textit{(KL)} \qquad R(\pi_{\mathbb{Q}}) \leqslant_{1-\delta} \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}) + \frac{1}{p_{\min}} \sqrt{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{n+1}{\delta}}{2n}} \tag{5}$$

$$\textit{(Bernstein)} \qquad R(\pi_{\mathbb{Q}}) \leqslant_{1-\delta} \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}) + O\Big(\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{1}{\delta}}{n}\Big) \qquad \textit{if } \mathbb{Q} \textit{ satisfies Eq.17} \tag{6}$$

$$R(\pi_{\mathbb{Q}}) \leqslant_{1-\delta} \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}) + O\Big(\sqrt{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{1}{\delta}}{n p_{\min}}}\Big) \qquad \textit{otherwise} \tag{7}$$

$$\textit{(Azuma-Hœffding)} \qquad R(\pi_{\mathbb{Q}}) \leqslant_{1-\delta} \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}) + O\Big(\sqrt{\frac{p_{\min}^2 \mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2}{\delta}}{n}}\Big). \tag{8}$$

4.2 Outline of proof

The complete proof is provided in Appendix B. Let $\mathcal{D}$ and $\mathcal{D}_n$ respectively denote the true and empirical joint distributions of context x and reward function ρ. The idea consists in constructing empirical counterfactual risk functions $r_{\mathcal{D}}$ and $r_{\mathcal{D}_n}$ such that distributions $\mathbb{P}$, $\mathbb{Q}$ over deterministic hypothesis space $\mathcal{H}$ can be applied to them:

$$r_{\mathcal{D}}(h) = \left\langle \mathcal{D}, 1 - \mathbb{E}_{(x_i,\rho_i)\sim\mathcal{D}} \mathbb{E}_{k_i \sim \pi_{\mathbb{P}}(x_i)} \frac{\mathbb{I}\{h(c_i) = k_i\}\rho_i(x_i,k_i)}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i,k_i)\}} \right\rangle \tag{9}$$

$$r_{\mathcal{D}_n}(h) = \left\langle \mathcal{D}_n, 1 - \mathbb{E}_{(x_i,\rho_i)\sim\mathcal{D}} \mathbb{E}_{k_i \sim \pi_{\mathbb{P}}(x_i)} \frac{\mathbb{I}\{h(x_i) = k_i\}\rho_i(x_i,k_i)}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i,k_i)\}} \right\rangle. \tag{10}$$

Recall that we can express the true and estimated CIS risk for $\pi_{\mathbb{Q}}$ as

$$R_{\mathrm{CIS}}(\pi_{\mathbb{Q}}) = \langle\, \mathbb{Q}, r_{\mathcal{D}} \rangle \qquad \text{and} \qquad \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \tau_n^{\pi_{\mathbb{P}}}) = \langle\, \mathbb{Q}, r_{\mathcal{D}_n} \rangle \tag{11}$$

and then use that $\dfrac{1}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i,k_i)\}} \leqslant \dfrac{1}{\pi_{\mathbb{P}}(x_i,k_i)}$ in order to obtain that $R_{\mathrm{CIS}}(\pi_{\mathbb{Q}}) \geqslant R(\pi_{\mathbb{Q}})$ and

$$R(\pi_{\mathbb{Q}}) - \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\pi_{\mathbb{P}}}) \leqslant R_{\mathrm{CIS}}(\pi_{\mathbb{Q}}) - \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\pi_{\mathbb{P}}}) = \langle\, \mathbb{Q}, r_{\mathcal{D}} \rangle - \langle\, \mathbb{Q}, r_{\mathcal{D}_n} \rangle. \tag{12}$$

Theorem 4.2 is then obtained by applying various inequalities to bound the right side of the following where $(x_i, k_i) \in \mathcal{F}_n^{\mathbb{P}}$:

$$n\langle\, \mathbb{Q}, r_{\mathcal{D}} - r_{\mathcal{D}_n} \rangle = \left\langle \mathbb{Q}, \sum_{i=1}^{n} \left[r_{\mathcal{D}} - \Big(1 - \mathbb{E}_{h\in\mathcal{H}} \frac{\mathbb{I}\{h(x_i) = k_i\}\rho(x_i,k_i)}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i,k_i)\}}\Big) \right] \right\rangle \tag{13}$$

and dividing by n.

4.3 Discussion

Using the PAC-Bayes-KL, PAC-Bayes-Bernstein and PAC-Bayes-Azuma-Hœffding bounds for an arbitrary martingale process allows to provide tight bounds on the true counterfactual risk of a contextual bandit. One observes that the previous result (Theorem 4.1) originally proposed by London and Sandler (2018) is $\mathcal{O}\Big(\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{n}{\delta}}{p_{\min} n}\Big)$. In comparison, the proposed result offers the following gains:

- the proposed KL-based bound (Eq. 5) tightens at a faster rate $1/\sqrt{n}$;
- the Azuma-Hœffding-based bound (Eq. 8) saves a rate $1/p_{\min}$ and tightens at a faster rate $1/\sqrt{n}$;
- the Bernstein-based bound (Eq. 6 and 7) saves a rate $1/p_{\min}$ if $\mathbb{Q}$ satisfies the condition of Eq. 17, otherwise it saves a rate $1/\sqrt{p_{\min}}$ and tightens at a faster rate $1/\sqrt{n}$.

In other words, all three bounds are more efficient than the existing result, in the sense that they either converge faster (better dependence on n) or they are less impacted by the clipping factor (better dependence on $p_{\min}$). A detailed argument outlining dominance of Eq. 5 by Theorem 4.1 under a looser condition on $\hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}})$ can be found in the appendix.

5 Conclusion

We derived three bounds on the true counterfactual risk within the contextual bandit setting based on PAC-Bayes-KL, PAC-Bayes-Bersntein and PAC-Bayes-Azuma-Hœffding inequalities, which improve upon existing results (London and Sandler, 2018). A natural line of extension would be to study the efficiency of PAC-Bayes bounds in the sequential decision making setting as applied to offline (Mandel et al., 2016) or doubly robust off-policy (Farajtabar et al., 2018) policy evaluation.

References

R. Bhatia and C. Davis. A better bound on the variance. *The American Mathematical Monthly*, 107(4):353–357, 2000.

M. Dudík, J. Langford, and L. Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011.

M. Farajtabar, Y. Chow, and M. Ghavamzadeh. More robust doubly robust off-policy evaluation. *arXiv preprint arXiv:1802.03493*, 2018.

N. Hassanpour and R. Greiner. A novel evaluation methodology for assessing off-policy learning methods in contextual bandits. In *Advances in Artificial Intelligence: 31st Canadian Conference on Artificial Intelligence, Canadian AI 2018, Toronto, ON, Canada, May 8–11, 2018, Proceedings 31*, pages 31–44. Springer, 2018.

M. J. Kearns, Y. Mansour, and A. Y. Ng. Approximate planning in large pomdps via reusable trajectories. In *Advances in Neural Information Processing Systems*, pages 1001–1007, 2000.

J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2008.

L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, 2011.

L. Li, W. Chu, J. Langford, T. Moon, and X. Wang. An unbiased offline evaluation of contextual bandit algorithms with generalized linear models. In *Proceedings of the Workshop on On-line Trading of Exploration and Exploitation 2*, 2012.

B. London and T. Sandler. Bayesian counterfactual risk minimization. *arXiv*, abs/1806.11500, 2018.

T. Mandel, Y.-E. Liu, E. Brunskill, and Z. Popović. Offline evaluation of online reinforcement learning algorithms. In *AAAI*, 2016.

J. Mary, P. Preux, and O. Nicol. Improving offline evaluation of contextual bandit algorithms via bootstrapping techniques. In *ICML*, 2014.

D. McAllester. Simplified pac-bayesian margin bounds. In *Learning theory and Kernel machines*, pages 203–215. Springer, 2003.

D. A. McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

D. Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

M. Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *Journal of machine learning research*, 3(Oct):233–269, 2002.

Y. Seldin, P. Auer, J. S. Shawe-Taylor, R. Ortner, and F. Laviolette. Pac-bayesian analysis of contextual bandits. In *NIPS*, 2011.

Y. Seldin, F. Laviolette, N. Cesa-Bianchi, J. Shawe-Taylor, and P. Auer. Pac-bayesian inequalities for martingales. *IEEE Transactions on Information Theory*, 58(12):7086–7093, 2012.

J. Shawe-Taylor and R. C. Williamson. A pac analysis of a bayesian estimator. In *Annual Workshop on Computational Learning Theory: Proceedings of the tenth annual conference on Computational learning theory*, volume 6, pages 2–9, 1997.

J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44(5):1926–1940, 1998.

A. Swaminathan and T. Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *ICML*, 2015.

I. O. Tolstikhin and Y. Seldin. Pac-bayes-empirical-bernstein inequality. In *NIPS*, 2013.

L. G. Valiant. A theory of the learnable. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445. ACM, 1984.

A Appendix

Theorem A.1 (PAC-Bayes-KL Inequality (Seeger, 2002))**.** *Let $\mathcal{H}$ be a hypothesis space, let $\overline{Z}_1, \ldots, \overline{Z}_n$ be a sequence of random functions, such that $\overline{Z}_i : \mathcal{H} \to [0,1]$ for $i = 1, \ldots, n$. Assume $\mathbb{E}[\overline{Z}_i | \overline{Z}_1 \ldots \overline{Z}_{i-1}] = \overline{b}$, where $\overline{b} : \mathcal{H} \to [0,1]$ is a deterministic function.*

Let $\overline{S}_n = \sum_{i=1}^n \overline{Z}_i$. Fix a prior distribution $\mathbb{P}$ over $\mathcal{H}$. Then for any $\delta \in (0,1)$, with probability greater than $1-\delta$ over $\overline{Z}_1 \ldots \overline{Z}_n$, for all distributions $\mathbb{Q}$ over $\mathcal{H}$ simultaneously:

$$\mathrm{KL}\left(\langle \frac{1}{n}\overline{S}_n, \mathbb{Q}\rangle || \langle \overline{b}, \mathbb{Q}\rangle\right) \leqslant \frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{n+1}{\delta}}{n}, \tag{14}$$

which is tight if $\langle \frac{1}{n}\overline{S}_n, \mathbb{Q}\rangle$ is close to zero or one, otherwise

$$\left| \langle \frac{1}{n}\overline{S}_n, \mathbb{Q}\rangle - \langle \overline{b}, \mathbb{Q}\rangle \right| \leqslant \sqrt{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{n+1}{\delta}}{2n}} \tag{15}$$

is tighter.

Theorem A.2 (PAC-Bayes-Azuma-Hœffding Inequality (Seldin et al., 2012))**.** *Let $\mathcal{H}$ be a hypothesis space, let $\overline{Z}_1, \ldots, \overline{Z}_n$ be a sequence of random functions, such that $\overline{Z}_i : \mathcal{H} \to [\alpha_i, \beta_i]$ where $\alpha_i, \beta_i \in \mathbb{R}$ for $i = 1, \ldots, n$ and pick $c > 1$. Let $\overline{M}_i = \sum_{j=1}^i \overline{Z}_j$. Fix a prior distribution $\mathbb{P}$ over $\mathcal{H}$. Then for any $\delta \in (0,1)$, with probability greater than $1-\delta$ over $\overline{Z}_1 \ldots \overline{Z}_n$, for all distributions $\mathbb{Q}$ over $\mathcal{H}$ simultaneously:*

$$|\langle \mathbb{Q}, \overline{M}_n\rangle| \leqslant_{1-\delta} \frac{1+c}{2\sqrt{2}} \sqrt{\left(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2}{\delta} + \varepsilon(\mathbb{Q})\right) \sum_{i=1}^n (\beta_i - \alpha_i)^2} \tag{16}$$

where

$$\varepsilon(\mathbb{Q}) = \frac{\ln 2}{2 \ln c}\left(1 + \ln\left\{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P})}{\ln \frac{2}{\delta}}\right\}\right).$$

Theorem A.3 (PAC-Bayes-Bernstein Inequality (Tolstikhin and Seldin, 2013))**.** *Let $\mathcal{H}$ be a hypothesis space, let $\overline{Z}_1, \ldots, \overline{Z}_n$ be a sequence of random functions, such that $\overline{Z}_i : \mathcal{H} \to \mathbb{R}$. Assume $\mathbb{E}[\overline{Z}_i | \overline{Z}_1 \ldots \overline{Z}_{i-1}] = \overline{0}$. Thus $\forall h \in \mathcal{H}$, $\overline{Z_1}(h), \ldots, \overline{Z_n}(h)$ is a martingale difference sequence. Let $\overline{M}_i = \sum_{j=1}^i \overline{Z}_j$ and hence $\mathbb{E}[\overline{M}_{i+1} | \overline{M}_1 \ldots \overline{M}_i] = \overline{M}_i$. Then $\forall h \in \mathcal{H}$, $\overline{M}_1(h), \ldots, \overline{M}_n(h)$ is a martingale. Let $\overline{V}_i : \mathcal{H} \to \mathbb{R}$ be such that $\overline{V}_i(h) = \sum_{j=1}^i \mathbb{E}[\overline{Z}_j(h)^2 | \overline{Z}_1(h), \ldots, \overline{Z}_{j-1}(h)]$. Assume that $||\overline{Z}_i||_\infty \leqslant K$ $\forall i$ with probability 1 and pick $\lambda \leqslant \frac{1}{K}$. Fix a prior distribution $\mathbb{P}$ over $\mathcal{H}$ and pick $c > 1$. Then for any $\delta \in (0,1)$, with probability greater than $1-\delta$ over $\overline{Z}_1 \ldots \overline{Z}_n$, for all distributions $\mathbb{Q}$ over $\mathcal{H}$ simultaneously which satisfy:*

$$\sqrt{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2v}{\delta}}{(e-2)\langle \mathbb{Q}, \overline{V}_n\rangle}} \leqslant \frac{1}{K} \tag{17}$$

the following holds:

$$|\langle \mathbb{Q}, \overline{M}_n\rangle| \leqslant (1+c)\sqrt{(e-2)\langle \mathbb{Q}, \overline{V}_n\rangle (\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2v}{\delta})} \tag{18}$$

where

$$v = \left\lceil \frac{\ln\left(\sqrt{\frac{(e-2)n}{\ln(\frac{2}{\delta})}}\right)}{\ln(c)} \right\rceil + 1,$$

and for all other $\mathbb{Q}$:

$$|\langle \mathbb{Q}, \overline{M}_n\rangle| \leqslant 2K(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2v}{\delta}). \tag{19}$$

Theorem A.4 (Bhatia-Davis Inequality (Bhatia and Davis, 2000))**.** *Let $\mathbb{P}$ be a distribution with support $(m, M) \subseteq \mathbb{R}$ and $\mathbb{E}[\mathbb{P}] = \mu$. Then, the following holds:*

$$\mathbb{V}[\mathbb{P}] \leqslant (M - \mu)(\mu - m). \tag{20}$$

Tightness of PAC-Bayes-KL extension. To show that 4.1 is looser than (28) is equivalent to showing that there exists an $N > 0$ such that, for all $n > N$,

$$\sqrt{\frac{2(\frac{1}{p_{\min}} - 1 + \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}))(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n}{\delta})}{p_{\min}(n-1)}} + \frac{2(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n}{\delta})}{p_{\min}(n-1)} - \sqrt{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n+1}{\delta}}{p_{\min}^2 2n}} \geqslant 0.$$

Note that for $n > 1$, the second term is non-negative by properties of KL divergence and the fact that, for $\delta \in (0, 1)$ and $n \geqslant 1$, $\ln n - \ln\delta > 0$.

$$\sqrt{\frac{4n(n-1)(1 + \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}})p_{\min} - p_{\min})(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n}{\delta})}{p_{\min}^2(n-1)^2 2n}} - \sqrt{\frac{(n-1)^2(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n+1}{\delta})}{p_{\min}^2(n-1)^2 2n}}$$

Completing the difference of squares and getting rid of the common denominator yields

$$4n(n-1)(1 + \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}})p_{\min} - p_{\min})(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n}{\delta}) - (n-1)^2(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n+1}{\delta})$$

Factoring out terms dependent on sample complexity yields

$$4n(n-1)\ln n - (n-1)^2\ln(n+1) = (n-1)(3n+1)\ln n + 1 \geqslant 0$$

for $n \geqslant 1$ and $\hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\mathbb{P}}) = \frac{1}{p_{\min}}(c-1)$ where $c \geqslant \frac{(n-1)^2\ln(n+1)}{4n(n-1)\ln n}$. Therefore, picking N large enough ensures a tighter bound in the limit and completes the argument. □

B Detailed proof of Theorem 4.2

Proof. Let $\mathcal{D}$ and $\mathcal{D}_n$ respectively denote the true and empirical joint distributions of context x and reward function ρ. We construct empirical counterfactual risk functions $r_{\mathcal{D}}$ and $r_{\mathcal{D}_n}$ such that distributions $\mathbb{P}$, $\mathbb{Q}$ over deterministic hypothesis space $\mathcal{H}$ can be applied to them:

$$r_{\mathcal{D}}(h) = \left\langle \mathcal{D}, 1 - \mathbb{E}_{(x_i,\rho_i)\sim\mathcal{D}}\mathbb{E}_{k_i\sim\pi_{\mathbb{P}}(x_i)}\frac{\mathbb{I}\{h(c_i) = k_i\}\rho_i(x_i,k_i)}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i,k_i)\}}\right\rangle \tag{21}$$

$$r_{\mathcal{D}_n}(h) = \left\langle \mathcal{D}_n, 1 - \mathbb{E}_{(x_i,\rho_i)\sim\mathcal{D}}\mathbb{E}_{k_i\sim\pi_{\mathbb{P}}(x_i)}\frac{\mathbb{I}\{h(x_i) = k_i\}\rho_i(x_i,k_i)}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i,k_i)\}}\right\rangle \tag{22}$$

. (23)

We can express the true and estimated CIS risk for $\pi_{\mathbb{Q}}$ as

$$R_{\mathrm{CIS}}(\pi_{\mathbb{Q}}) = \langle\, \mathbb{Q}, r_{\mathcal{D}}\rangle \qquad \text{and} \qquad \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \tau_n^{\pi_{\mathbb{P}}}) = \langle\, \mathbb{Q}, r_{\mathcal{D}_n}\rangle \tag{24}$$

then use that $\dfrac{1}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i,k_i)\}} \leqslant \dfrac{1}{\pi_{\mathbb{P}}(x_i,k_i)}$ to obtain that $R_{\mathrm{CIS}}(\pi_{\mathbb{Q}}) \geqslant R(\pi_{\mathbb{Q}})$ and

$$R(\pi_{\mathbb{Q}}) - \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\pi_{\mathbb{P}}}) \leqslant R_{\mathrm{CIS}}(\pi_{\mathbb{Q}}) - \hat{R}_{\mathrm{CIS}}(\pi_{\mathbb{Q}}, \mathcal{F}_n^{\pi_{\mathbb{P}}}) = \langle\, \mathbb{Q}, r_{\mathcal{D}}\rangle - \langle\, \mathbb{Q}, r_{\mathcal{D}_n}\rangle. \tag{25}$$

We will then apply different inequalities that bound the right side of the following with $(x_i, k_i) \in \mathcal{F}_n^{\mathbb{P}}$:

$$n\langle\, \mathbb{Q}, r_{\mathcal{D}} - r_{\mathcal{D}_n}\rangle = \left\langle \mathbb{Q}, \sum_{i=1}^{n}\left[r_{\mathcal{D}} - \left(1 - \mathbb{E}_{h\in\mathcal{H}}\frac{\mathbb{I}\{h(x_i) = k_i\}\rho(x_i,k_i)}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i,k_i)\}}\right)\right]\right\rangle. \tag{26}$$

Using PAC-Bayes-KL inequality (Theorem A.1) Scaling r_D and $1 - \frac{\mathbb{I}\{h(x_i)=k_i\}\rho(x_i,k_i)}{\max\{p_{\min},\pi_{\mathbb{P}}(x_i,k_i)\}}$ to $[0,1]$, constant $(p_{\min} - 1)$ addition and subtraction cancel out, we have

$$\langle\, \mathbb{Q}, p_{\min}(r_{\mathcal{D}} - r_{\mathcal{D}_n})\rangle \leqslant_{1-\delta} \sqrt{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n+1}{\delta}}{2n}} \tag{27}$$

$$\text{thus } \langle\, \mathbb{Q}, (r_{\mathcal{D}} - r_{\mathcal{D}_n})\rangle \leqslant_{1-\delta} \frac{1}{p_{\min}}\sqrt{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln\frac{n+1}{\delta}}{2n}}. \tag{28}$$

Using PAC-Bayes-Azuma-Hœffding inequality (Theorem A.2) For any $c > 1$, we have

$$n\langle \mathbb{Q}, r_{\mathcal{D}} - r_{\mathcal{D}_n} \rangle \leqslant_{1-\delta} \frac{1+c}{2\sqrt{2}} \sqrt{\left(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2}{\delta} + \frac{\ln 2}{2 \ln c} \Big(1 + \ln \big(\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P})}{\ln \frac{2}{\delta}} \big) \Big) \right) \frac{n}{p_{\min}^2}}. \tag{29}$$

Using PAC-Bayes-Bernstein inequality (Theorem A.3) For any $\lambda > 0$, we have

$$n\langle \mathbb{Q}, r_{\mathcal{D}} - r_{\mathcal{D}_n} \rangle \leqslant_{1-\delta} \frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2}{\delta}}{\lambda} + (e-2)\lambda \langle \mathbb{Q}, \overline{V}_n \rangle \tag{30}$$

$$\text{where } \overline{V}_i(h) = \sum_{j=1}^{i} \mathbb{E}[\overline{Z}_j(h)^2 | \overline{Z}_1(h), \ldots, \overline{Z}_{j-1}(h)]$$

$$\text{and } \overline{Z}_i(h) = r_{\mathcal{D}} - \Big(1 - \frac{\mathbb{I}\{h(x_i) = k_i\} \rho(x_i, k_i)}{\max\{p_{\min}, \pi_{\mathbb{P}}(x_i, k_i)\}} \Big) \qquad \text{noting that } ||\overline{Z}_i|| \leqslant K = \frac{1}{p_{\min}}.$$

For $\mathbb{Q}$ that satisfies $\sqrt{\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2v}{\delta}}{(e-2)\langle \mathbb{Q}, \overline{V}_n \rangle}} \leqslant \frac{1}{K} = p_{\min}$, we have that for any $c > 1$,

$$n\langle \mathbb{Q}, r_{\mathcal{D}} - r_{\mathcal{D}_n} \rangle \leqslant_{1-\delta} (1+c) \sqrt{(e-2)\langle \mathbb{Q}, \overline{V}_n \rangle \Big(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{2v}{\delta} \Big)} \qquad \text{where } v = \left\lceil \frac{\ln \frac{(e-2)n}{\ln \frac{2}{\delta}}}{2 \ln c} \right\rceil + 1. \tag{31}$$

By using the facts that $\overline{Z}_i \in [-1, \frac{1}{p_{\min}}]$ and $\mathbb{E}[\overline{Z}_j | \overline{Z}_1, \ldots, \overline{Z}_{j-1}] = \overline{0}$ combined with the Bhatia-Davis inequality (Theorem A.4, Bhatia and Davis (2000)), we can bound

$$\mathbb{E}[\overline{Z}_n(j)^2 | \overline{Z}_1(h), \ldots, \overline{Z}_{j-1}(h)] \leqslant \Big(\frac{1}{p_{\min}} - 0 \Big) \Big(0 - (-1) \Big) = \frac{1}{p_{\min}} \quad \text{s.t.} \quad \overline{V}_n \leqslant \frac{n}{p_{\min}}. \tag{32}$$

Thus,

$$n\langle \mathbb{Q}, r_{\mathcal{D}} - r_{\mathcal{D}_n} \rangle \leqslant O\Bigg(\sqrt{\langle \mathbb{Q}, \overline{V}_n \rangle \Big(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{1}{\delta} \Big)} \Bigg) \leqslant O\Bigg(\sqrt{n \Big(\frac{\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \frac{1}{\delta}}{p_{\min}} \Big)} \Bigg). \tag{33}$$

For all other $\mathbb{Q}$:

$$n\langle \mathbb{Q}, r_{\mathcal{D}} - r_{\mathcal{D}_n} \rangle \leqslant_{1-\delta} 2K \Bigg(\mathrm{KL}(\mathbb{Q}||\mathbb{P}) + \ln \Big(\frac{2v}{\delta} \Big) \Bigg). \tag{34}$$

□

MinAtar: An Atari-inspired Testbed for More Efficient Reinforcement Learning Experiments

Kenny Young
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
kjyoung@ualberta.ca

Tian Tian
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
ttian@ualberta.ca

Abstract

The Arcade Learning Environment (ALE) is a popular platform for evaluating reinforcement learning agents. Much of the appeal comes from the fact that Atari games are varied, showcase aspects of competency we expect from an intelligent agent, and are not biased towards any particular solution approach. The challenge of the ALE includes 1) the representation learning problem of extracting pertinent information from the raw pixels, and 2) the behavioural learning problem of leveraging complex, delayed associations between actions and rewards. In many cases, the research questions we are interested in pertain more to the latter, but the representation learning problem adds significant computational expense. In response, we introduce MinAtar, short for miniature Atari, a new evaluation platform that captures the general mechanics of specific Atari games, while simplifying certain aspects. In particular, we reduce the representational complexity to focus more on the behavioural challenges. MinAtar consists of analogues to five Atari games which play out on a 10x10 grid. MinAtar provides the agent with a 10x10xn state representation. The n channels correspond to game-specific objects, such as ball, paddle and brick in the game Breakout. While significantly simplified, these domains are still rich enough to allow for interesting behaviours, similar to those observed in the ALE. To demonstrate the challenges posed by these domains, we evaluated a smaller version of the DQN architecture. We also tried variants of DQN without experience replay, and without a target network, to assess the impact of those two prominent components in the MinAtar environments. In addition, we evaluated a simpler agent that used actor-critic with eligibility traces, online updating, and no experience replay. We hope that by introducing a set of simplified, Atari-like games we can allow researchers to more efficiently investigate the unique behavioural challenges provided by the ALE.

Keywords: Reinforcement Learning, Evaluation Environment

Acknowledgements

The authors would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and Alberta Innovates. We would also like to thank Martha White, Andy Patterson and the rest of the University Alberta RLAI group for helpful comments and feedback.

1 Motivation

The arcade learning environment (Bellemare, Naddaf, Veness, & Bowling, 2013) (ALE) has become widely popular as a testbed for reinforcement learning (RL), and other AI algorithms. An important aspect of the ALE's appeal is that the environments are designed to be interesting for human players, and not to be amenable to any particular approach to AI. Because of this design, the platform is largely free of experimenter bias and provides diverse challenges which we associated with the kind of general intelligence seen in humans.

The challenges provided by the ALE can be broadly divided into two aspects: 1) the representation learning problem of extracting pertinent information from the raw pixels, and 2) the behavioural learning problem of leveraging complex, delayed associations between actions and rewards.

While it is important to have testbeds that provide this kind of broad-spectrum challenge, it is not always what we want as experimenters. Often, the work flow when evaluating a new RL idea is to first experiment with very simple domains, such as *Mountain Car* or a tabular MDP, then jump to complex domains, like those provided by the ALE, to validate the intuition. We believe that this jump tends to leave a wide gap in understanding that would be best filled by domains of intermediate complexity.

MinAtar is intended to bridge this gap by providing environments designed to capture the spirit of specific Atari 2600 games, while simplifying certain aspects. One aspect of the ALE that makes it difficult to use as an RL testbed is that much of the computing power expended to train a deep RL agent goes toward learning a semantically meaningful representation from the raw pixel input. When the first deep RL agents were shown to succeed in the ALE, this was an interesting challenge. Presently, however, this challenge is usually addressed with some variant of convolutional neural network and often the interesting research questions come not from this visual representation learning problem, but instead from the higher level behavioural challenges involved in the various games. A major goal of MinAtar is, thus, to reduce the complexity of this representation learning problem while maintaining the mechanics of the original games as much as possible. While our simplification also reduces the behavioural complexity of the games, the MinAtar environments are still rich enough to showcase interesting behaviours, similar to those observed in the ALE.

We emphasize that MinAtar is not a challenge problem, like Go, StarCraft (Vinyals et al., 2017) or the ALE when it was first introduced. The purpose is to serve as a more efficient way to validate intuition, and provide proof of concept for artificial intelligence ideas, which is closer to how the ALE is often used today.

2 The MinAtar Platform

Aside from replicating the spirit of a set of Atari 2600 games, the design goals of MinAtar can be broken down as follows:

- **Reduce spatial dimension:** In MinAtar, each game takes place on a 10x10 grid. This is a significant reduction from the Atari 2600 screen size of 160x210. Often, in the ALE, the input to the learning agent is down-sampled. For example, Mnih et al. (2015) downsample to 64x64. MinAtar provides a much smaller input without the need for this step.
- **Reduce action space:** In MinAtar, the action space consists of moving in one of the 4 cardinal directions, firing, or no-op. This makes for a total of just 6 actions. On the other hand, in the ALE, it is possible to move in 8 directions or stand still. For each of these choices, the player can also either fire or not fire. This makes for a total of 18 actions.
- **Provide semantically meaningful input:** Instead of raw color channels, each MinAtar environment provides a number of semantically meaningful channels. For example, for the game *Breakout*, MinAtar provides channels for *ball*, *paddle* and *brick*. The total number of such channels is game-dependent. The state provided to the agent consists of a stack of 10x10 grids, one for each channel, giving a total dimensionality of 10x10xn where n is the number of channels.
- **Reduce partial observability:** Many games in the ALE involve some benign form of partial observability. For example, the motion direction of objects is often not discernible from a single frame. Techniques like frame stacking (Mnih et al., 2015) reduce such partial observability. Minitar mitigates the need for such techniques by making the motion direction of objects discernible within a single frame. Depending on the situation, we convey motion direction either by providing a *trail* channel indicating the last location of certain objects, or by explicitly providing a channel for each possible direction of motion. We do not aim to eliminate partial observability, but merely mitigate the more trivial instances.
- **Simplify certain game mechanics:** Reduction to a 10x10 grid means that some of the more nuanced mechanics of certain Atari 2600 games are difficult or impossible to replicate. Other mechanics we left out for simplicity. For example, in *Space Invaders* we do not include the destructible defence bunkers or the mystery ship which

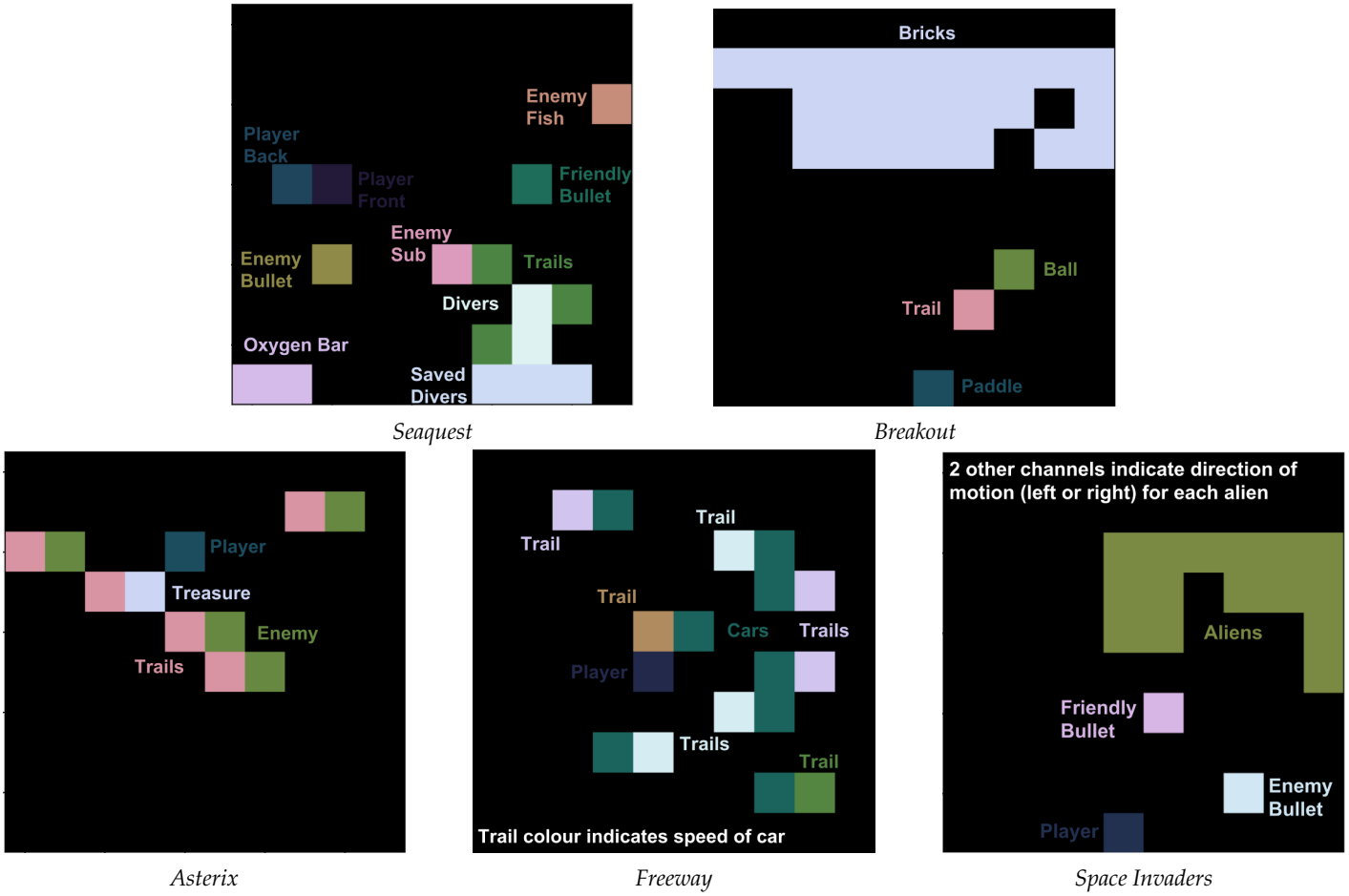
periodically crosses the top of the screen. We also limit each game to one life, terminating as soon as the agent dies.

- **Add stochasticity:** The Atari 2600 is deterministic. Each game begins in a unique start state and the outcome is uniquely determined by the action sequence that follows. This deterministic behaviour can be exploited by simply repeating specific sequences of actions, rather than learning policies that generalize. Machado et al. (2017) address this by adding *sticky-actions*, where the environment repeats the last action with probability 0.25 instead of executing the agent's current action. We incorporate sticky-actions in MinAtar, but with a smaller probability of 0.1. This is based on the assumption that individual actions have a larger impact in MinAtar than in the ALE due to the larger granularity of the movement discritization, thus each sticky-action can have a potentially larger negative impact. In addition, we make the spawn location of certain entities random. For example, in *Seaquest* the enemy fish, enemy submarines and divers emerge from random locations on the side of the screen.

So far, we have implemented five games for the MinAtar platform. Visualizations of each of these games are shown in Figure 1. MinAtar is available as an open-source python library under the terms of the GNU General Public License. The source code is available at:

https://github.com/kenjyoung/MinAtar

You can find links to videos of trained DQN agents playing the MinAtar games in the README.



Seaquest *Breakout*

Asterix *Freeway* *Space Invaders*

Figure 1: Visualization of each MinAtar game. Colour indicates active channel at each spatial location, but note that the representation provided by the environment consists of binary values for each channel and not RGB values.

3 Experiments

We provide results for several variants of two main algorithms on each of the five MinAtar environments. We trained each agent for a total of 5 million frames, compared to 50 million for Mnih et al. (2015) (200 million if you count in terms of

emulator frames since they use frame-skipping). The reduction in the number of frames allowed us to run more repeats without inordinate expense. We were able to train 30 different random-seeds per agent-environment combination to obtain results with tighter confidence intervals. These results serve to provide a baseline for future work, as well as to illustrate the challenge posed by these new environments.

Deep Q-Network

Our DQN architecture consisted of a single convolutional layer, followed by a fully connected hidden layer. Our convolutional layer used 16 3x3 convolutions with stride 1, while our fully connected layer had 128 units. 16 and 128 were chosen as one quarter of the final convolutional layer and fully connected hidden layer respectively of Mnih et al. (2015). We also reduced the replay buffer size, target network update frequency, epsilon annealing time and replay buffer fill time, each by a factor of ten relative to Mnih et al. (2015) based on the reasoning that our environments take fewer frames to master than the original Atari games. We trained on every frame and did not employ frame skipping. The reasoning behind this decision is that each frame of our environments is more information rich. Other hyperparameters, including the step-size parameter, were set to match Mnih et al. (2015). We also tested variants of the DQN architecture without experience replay and without a target network to assess the usefulness of these components in the simpler environments.

The smaller architecture and input size means that running on CPU instead of GPU was feasible. For DQN with experience replay running on *Seaquest*, the total wall-clock time per frame was around 8 milliseconds when running on a single CPU, compared to 5 milliseconds when running on GPU. We report these times for *Seaquest* because it has the largest number of input channels and thus the largest number of network parameters.

Actor-Critic with Eligibility Traces

We also experimented with an online actor-critic with eligibility traces (AC(λ)) agent (Degris, Pilarski, & Sutton, 2012; Sutton & Barto, 2018). This agent used no experience replay or multiple parallel actors. We used a similar architecture to the one used in our DQN experiments, except that we replaced the relu activation functions with the SiLU and dSiLU activation functions introduced by Elfwing, Uchibe, and Doya (2018). Their work showed these activations to be helpful when using online eligibility traces with nonlinear function approximation. Specifically, we applied SiLU in the convolutional layer and dSiLU in the fully connected hidden layer. We set the step-size to 2^{-8}, the largest value that was found to yield stable learning across games by Young, Wang, and Taylor (2018), who used online AC(λ) to train a similar architecture for several ALE games. We tested AC(λ) with trace decay parameter of 0.8 and 0, where the latter corresponds to one-step actor-critic with no eligibility trace.

Discussion

The results of our experiments are shown in Figure 2. The first thing to note is that the MinAtar environments clearly show the advantage of using experience replay in DQN. On the other hand, the target network appeared to have little performance impact. To verify that the poor performance of DQN without experience replay was not due to a poorly tuned step-size parameter, we tried running DQN without experience replay with various values of the step-size on *Seaquest*. We choose *Seaquest* for the step-size sweep because it showed the largest discrepancy in results with and without experience replay. Specifically, we tried 30 random-seeds with each value of the step-size from the set $\{\alpha_0 \cdot 2^i | i \in \{1, 0, ..., -4\}\}$, where the original value was $\alpha_0 = 0.00025$. We swept primarily lower values, reasoning that the lack of batching would lead to higher variance, potentially requiring a lower step-size relative to DQN with experience replay. For all the step-size choices, none of the average returns over the final 100 training episodes were above 1.0.

DQN significantly outperformed the simpler AC(λ) agent in 3 of the games. In each of these 3 games, AC(λ) barely learned at all. On the other hand, perhaps surprisingly, AC(λ) significantly outperformed DQN at *Space Invaders*. The two performed similarly at *Breakout*, though DQN converged faster to it's final performance. To verify that the AC(λ)'s poor performance was not due to a poorly tuned step-size, we performed a step-size sweep on *Seaquest* running AC(0.8). Specifically, we tried 30 random-seeds with each value of the step-size from the set $\{\alpha_0 \cdot 2^i | i \in \{-1, 0, ..., 4\}\}$, where $\alpha_0 = 2^{-8}$ is the original AC(λ) step-size. We swept primarily higher step-size values, reasoning that the initial step-size, chosen for stability on ALE games, was potentially unnecessarily low for some MinAtar games. With the best step-size of $\alpha_0 \cdot 2^3$, we observed performance improvement of AC(λ), achieving a final average return of 4.4 ± 0.6 over the final 100 episodes in 5 million training frames. However, this was still far below the performance of DQN with experience replay.

Taken together, these results suggest that the MinAtar environments are effective at highlighting the strengths and weaknesses of different approaches.

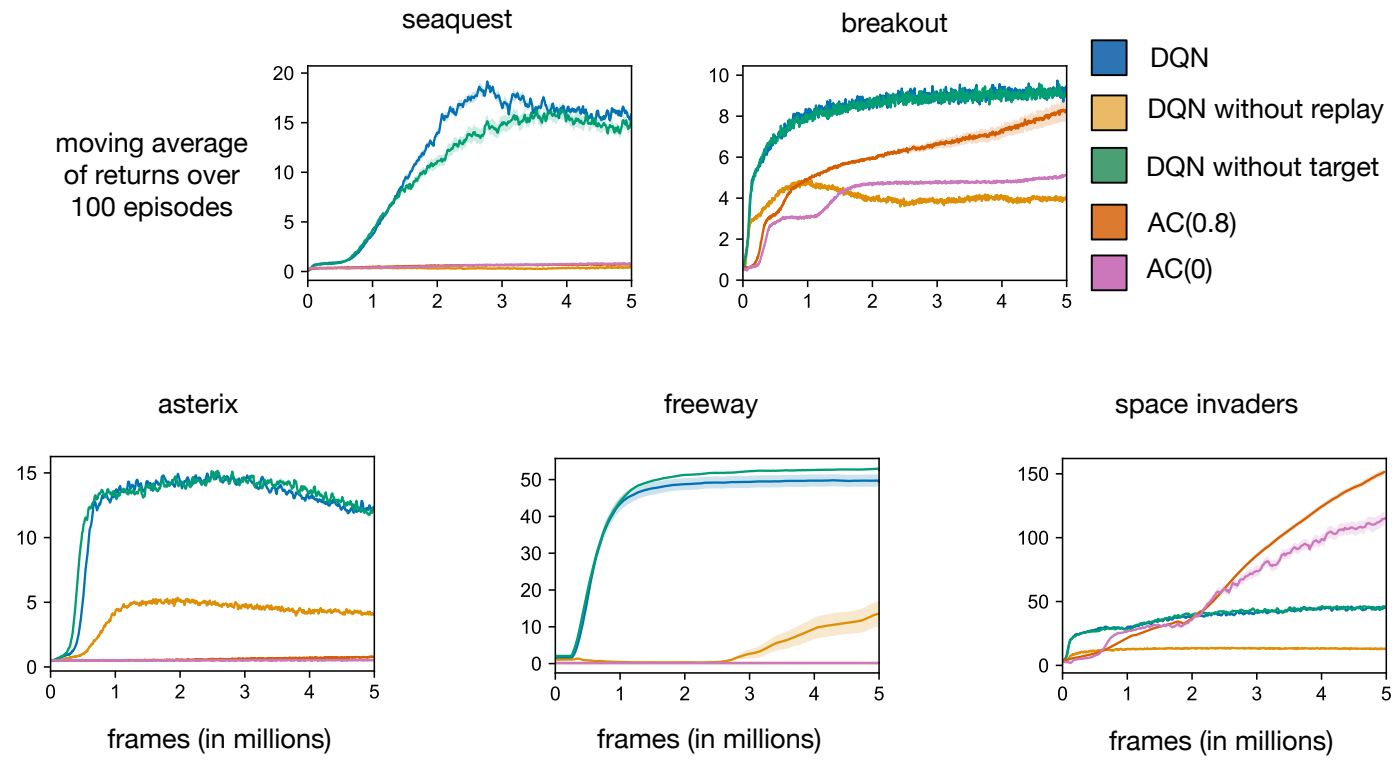


Figure 2: Average return v.s. training frames for all games and agents.

4 Conclusion

We introduce MinAtar, a new evaluation platform for reinforcement learning designed to allow for more efficient experiments by providing simpler versions of Atari 2600 games. These environments aim to reduce the representation learning burden to focus more on interesting behavioral aspects of the games, which are often of greater interest to RL experimenters. Currently the platform consists of five environments. In the future, we plan to add more. Of particular interest are the games typically considered hard exploration problems, such as *Montezuma's Revenge* and *Pitfall*. It would also be interesting to explore how other DQN additions, such as double DQN, perform in the MinAtar environments.

References

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253–279.

Degris, T., Pilarski, P. M., & Sutton, R. S. (2012). Model-free reinforcement learning with continuous action in practice. In *American control conference (acc), 2012* (pp. 2177–2182). IEEE.

Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, *107*, 3–11.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., & Bowling, M. (2017). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *arXiv preprint arXiv:1709.06009*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Sutton, R. S. & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., … Schrittwieser, J., et al. (2017). StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Young, K., Wang, B., & Taylor, M. E. (2018). Metatrace: Online step-size tuning by meta-gradient descent for reinforcement learning control. *arXiv preprint arXiv:1805.04514*.