Inferring the Intentions of Learning Agents

Author
VINCENT KUBALA

Advisor
AMY GREENWALD

Reader
MICHAEL LITTMAN

Abstract

This thesis addresses the problem of inferring the goals, represented as a utility function, of an intelligent agent who is learning in a known way. It generalizes the standard inverse reinforcement learning (IRL) problem, where it is assumed that the agent does not (need to) learn and makes decisions optimally. Here, it is assumed instead that the agent may learn in a known way and makes decisions in a known, albeit not necessarily optimal, way. In addition to laying out the general problem, I identify two of its special cases and solve them with polynomial-time algorithms. One formulates the problem as a linear program, and the other formulates the problem as one of maximum a posteriori estimation and uses a gradient descent approach that is guaranteed to converge.

Contents

1 A Motivating Story

Famous fictional friends Alice and Bob each have an unopened glass bottle of root beer. Alice observes Bob hold the bottle with his left hand, enclose his right hand on the bottle's cap, and pull up on the rim of the cap with his fingers. His brow line wrinkles, he clenches his jaw, and he makes a soft grunting noise, but to no effect. Suddenly, his behavior changes. He inserts the top of the bottle into his mouth.

"What is he doing?", wonders Alice, wanting to react appropriately. Alice, being the scientific (if sometimes lacking in social intuition) person that she is, brainstorms several hypotheses. Perhaps Bob is exercising! If so, Alice could offer suggestions for other exercises that Bob could try. That's possible, she concludes, but unlikely; there are tons of other ways to exercise, many of them better suited for the goal of getting in shape, so the chance that Bob chose this particular exercise is slim. Alternatively, maybe this is simply how Bob enjoys consuming glass bottles of root beer. Alice's prior knowledge leads her to doubt this hypothesis; she has never known anyone to enjoy themselves in this way before.

"Eureka!", Alice exclaims (to Bob's bewilderment): Bob is trying to *open* his bottle of root beer, and he doesn't know that the cap simply twists off. This explains all of Bob's behavior: he first attempted to pull the cap off with his fingers, and once he learned that that wasn't going to work, he changed his approach. Just as crucially, Alice can't imagine many other approaches that Bob could have taken to achieve this goal (besides twisting the cap, of course), and Alice knows this to be a common goal for people to have in this kind of situation.

"You're going to chip a tooth," she says. "The cap twists off." Bob easily twists the cap off and enjoys his root beer, his teeth unharmed.

In this story, Alice wanted to determine Bob's goals so that she could help him, but she had to overcome the challenge that Bob's behavior was neither optimal under his goal of opening the bottle (he did not simply twist off the cap) nor stationary (he tried two different approaches). This thesis formalizes and proposes solutions to Alice's problem.

2 Introduction

Most models of behavior assume that intelligent agents—people, animals, bacteria, robots—are motivated, meaning that, at any given time, they are trying to achieve some goal. These goals, however, are not directly observable. Nonetheless, another agent can learn something about them via (truthful) communication, or by observing their behavior. This thesis is concerned with the problem of learning an intelligent agent's motivations by observing their behavior.

One challenge in making this inference is that the recipe for turning a goal into behavior is *not* a function. Two agents with identical goals acting in identical environments might behave differently if, for example, they hold different beliefs. Two Netflix users with the exact same preferences over movies will likely end up watching different movies, because they are probably acting under different information about the movies from friends, reviews, and trailers. Two people whose only goal in using Instagram is to get the most "like"s possible may have different approaches to writing captions, because they may have differing beliefs about what earns them their likes. What's worse: differences in beliefs are difficult to take into account, because beliefs, like goals, are unobservable. Worse still, beliefs change over time, as an agent learns from their experiences in their environment.

Inverse reinforcement learning (IRL) is a broadly-studied model of the problem at hand in which an observer attempts to infer an actor's motivations by observing the actor's behavior. Two simplifying assumptions made in most IRL studies are: 1. the actor possesses perfect knowledge of his[1] (stationary) environment, and 2. the observer knows this about the actor. A relaxation of this problem, in which the actor's beliefs are unknown to the observer, but are still assumed to be fixed, can be solved by simultaneously inferring the goals and beliefs of the actor [Herman *et al.*, 2016]. This thesis addresses a different relaxation, in which the actor's beliefs change over time, and only his initial beliefs are known. To my knowledge, this is a new problem.

There are many reasons why this is a useful problem to solve. For one, by observing an agent who is learning, an observer can learn much more about his preferences than she can by observing an agent who is already an expert. A demonstration of this can be found in the appendix under the subsection, "The Fruit Machines". This suggests that observing young animals may be more useful in learning the fundamental values of a species than observing their more experienced parents. Additionally, it seems reasonable to expect that a well-featured representation of the actor's utility function can help to make decent predictions of the actor's behavior in a novel environment, whereas shallower models of the actor may be less transferable [Abbeel

[1]Throughout this paper, I adopt the convention of using male pronouns (he, him, his, etc.) for the actor and female pronouns for the observer, since this is consistent with the story about Alice and Bob and eases communication.

and Ng, 2004].

One of primary original motivations for IRL is *apprenticeship learning*, where the goal is to learn to replicate the behavior of an expert actor by learning and then acting upon his utility function. The more general problem with which this thesis is concerned may not apply to apprenticeship learning well, since it may not be desirable to learn from an actor who is still himself learning.[2] There is, however, an analogue of apprenticeship learning that this work intends to solve. Imagine that the *observer* is the expert in the relevant environment, and that the actor is blocked from achieving his goal, which is unknown to the observer, due to his incorrect beliefs or incomplete knowledge about the environment. This was the case with Alice and Bob; Bob was attempting to open his bottle of root beer, a goal that was initially unknown to Alice, but he was having trouble because he lacked a key piece of information about the environment that Alice knew: the cap twists off. Alice was able to learn Bob's goal and then prescribe an action to help him achieve it. I call this problem **mentorship learning**, to highlight its similarity to apprenticeship learning and also to evoke an image of a good mentor, who is an expert in her domain (e.g., academic research) but does not offer advice until she understands the goals (e.g., academic and career goals) of her mentee.[3] The two problems are similar in the sense that in each case, the observer is evaluated based on her ability to choose actions that maximize the actor's utility function. In the case of apprenticeship learning, this takes the form of the observer acting in the environment herself, and in mentorship learning, this takes the form of the observer prescribing actions to the actor. Mentorship learning is a useful problem to solve if we wish to create autonomous agents that help people accomplish their goals. A robot who solves Alice's problem and ones like it in particular may be useful to help elderly and disabled people gain independence, both in the physical and digital worlds.

Another use case of the work in this thesis is to infer the social values of an agent. The actor may interact with other intelligent agents in some environment, and his decisions may affect the other agents. In that case, the environment is an economic game. Traditional IRL is unfit for this setting, because even if the actor is an expert on the game, he is almost certainly not an expert on the behavior of all of the other agents, so he must learn how to best play with the other agents over time, perhaps via fictitious play [Boylan and El-Gamal, 1993] or, more generally, experience-weighted attraction learning [Camerer and Ho, 1999]. The methods presented here help to account for this, and they will be increasingly useful as answers to the question of behavioral game theory improve.

[2] though as an undergraduate student who loves to teach, I have a personal bias against this idea.

[3] Naturally, my image is of Professor Greenwald.

3 Formal Setup and Notation

The setup can be described informally as follows. There are two agents: an actor and an observer. The actor interacts with an environment, attempting to maximize some objective function, while the observer watches and attempts to infer that objective function based on the actor's decisions.

Now, I describe this more precisely. "The environment" comprises every relevant aspect of the world outside of the actor. Time advances in the environment in discrete units called **timesteps**. At each timestep t, the environment takes on exactly one **state** s_t, out of a finite set S of possible states. The actor observes s_t and then decides which **action** a_t to take, out of a finite set A of possible actions.[4] Once the actor has executed his action, the time increments and the environment advances to a potentially new state s_{t+1}. Upon the the actor's arrival to s_{t+1}, he earns an observable[5], **reward** r_{t+1}, which is a real number. $r_{t+1} = 0$ indicates the absence of a reward, so positive reward is a "good" reward, in the sense that it is better than nothing, and negative reward is a punishment or "bad reward", in the sense that it is worse than nothing.

A reward is something like a monetary payoff, points in a game, an amount of treats, or an extent of social praise. For those familiar with the distinction between intrinsic and extrinsic motivation [Benabou and Tirole, 2003], rewards provide extrinsic motivation, because they are a property of the environment in which the actor acts, and not of the actor.

As the actor interacts with the environment, an interaction history accumulates.

Definition 1. The **interaction history** through time t, written $\mathcal{H}^{(t)}$, is $(s_0, a_0, s_1, a_1, ..., s_t, a_t, s_{t+1})$.

3.1 Defining the Environment

In addition to the set S of states and the set A of actions, two objects define the environment: a description of how the environment changes state, and a description of how the environment rewards the actor:

Definition 2. The **transition function**, $T : S \times A \times S \to [0, 1]$, defines how the environment changes from one state to the next; it describes the environment's physics. In particular, $T(s, a, s')$ gives the probability with which the environment changes to state s', given that the actor took action a from the previous state s. Since T defines probability distributions, $\sum_{s' \in S} T(s, a, s') = 1$, for any $s \in S$ and $a \in A$.

[4] I assume for simplicity that the set of available actions does not depend on the current state.

[5] meaning that the observer observes it

Definition 3. The **reward function**, $R : S \to \mathbb{R}$, defines the rewards associated with each state. $R(s)$ gives the reward that the actor will earn by arriving in state $s \in S$.

Thus, the environment is a Markov Decision Process [Howard, 1960]:

Definition 4. A (finite) **Markov Decision Process (MDP)** is a tuple (S, A, T, R), where S, the set of states; A, the set of actions; T, the transition function; and R, the reward function, define an environment in the way described above.

3.2 Defining the Actor

Now, it must be understood that ***rewards do not have the final say in motivating the actor.*** Actors, often being people, have values that are idiosyncratic. Some people prefer the scenic route, whereas others want to get to their destination quickly. Some don't care too much to earn awards and gain membership into prestigious communities; others live for such accomplishments. Some are selfish, others altruistic. Some prefer social efficiency; others prefer equality. Along any of these spectra of values, different people desire to strike different balances.

The actor's true, final preferences, which account for *both* the reward function of the environment and his idiosyncrasies, are represented formally as a utility function[6] [Feldman and Serrano, 2006]:

Definition 5. The actor's **utility function** is a function $U : S \to \mathbb{R}$, such that the actor considers state s to be at least as good as state s' if and only if $U(s) \geq U(s')$.

As argued by Feldman and Serrano, this definition implies that the actor considers s to be strictly better than s' if and only if $U(s) > U(s')$ and that the actor is indifferent between s and s' if and only if $U(s) = U(s')$.

The actor, knowing his own utility function, aims to earn as much utility as possible. One wrinkle in stating the actor's goal precisely is that the utility function as defined above gives the utility of any one particular state but says nothing about how to compare different sequences of states. To state the actor's goal, it is necessary to first define the utility of a potentially infinite sequence of states. I iron out this wrinkle in the standard way, defining[7]:

$$U((s_0, s_1, ..., s_t, ...)) := \sum_t w_t U(s_t) \tag{1}$$

where $w_t \in [0, 1]$ is the weight on the utility of s_t, and $\sum_{t=0}^{\infty} w_t < \infty$. The overwhelmingly most common weighting scheme is exponential discounting, in which

[6]For the reader who is familiar with the distinction between objective and subjective reward [Austerweil *et al.*, 2015], "reward" here is objective reward, and "utility" here is subjective reward.

[7]overload U to also give the utility associated with a sequence of states

$w_t = \gamma^t$, where $\gamma \in [0, 1)$, so $\sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma} < \infty$, meaning that the utility associated with even an infinitely long sequence of states is finite, and the weight decreases with time at a consistent rate. However, there is evidence that other weighting schemes, such as the present-biased quasi-hyperbolic discounting[8] [Benhabib, J. *et al.*, 2010], are better at explaining some human behavior. The methods that I will present are compatible with quasi-hyperbolic discounting and, I imagine, most other weighting schemes for which one can write a learning algorithm.

Now, the actor's goal can be stated precisely as choosing which policy (see Definition 6) to execute in order to maximize Equation 1, for whatever time-weighting scheme he prefers.

Definition 6. A **policy** is a function $\pi : S \times A \rightarrow [0, 1]$, such that $\pi(s, a)$ is the probability with which the actor chooses action a when acting in state s. Overloading notation, let $\pi(s)$ be the vector whose kth element is $\pi(s, a_k)$.

To execute policy π is to choose an action at any state s by sampling from $\pi(s)$.

The black-box view of an actor is that he is simply a function that maps the interaction history through time t to the policy that he will execute at time t, given his (time-weighted) utility function. In this paper, I decompose this function into two separate parts, in exactly the way that [Camerer and Ho, 1999] does for learning in normal-form games. The parts are: 1) a **learning rule**, which learns an **attraction function** from the interaction history based on the utility function; and 2) a **decision rule**, which produces a policy from the attraction function. See Figure 1 for a visualization of the loop of the actor interacting with the environment. See Figure 2 for a more detailed diagram of how the actor determines a policy. Now we look at each piece of the system in more detail.

Definition 7. An **attraction function** is a function $Q^{(t)} : S \times A \rightarrow \mathbb{R}$ that incorporates both the actor's utility function and his accumulated knowledge about the environment from $\mathcal{H}^{(t)}$. $Q^{(t)}(s, a)$ gives the actor's "attraction" to taking action a from state s.

Loosely speaking, the actor should be more likely to take an action a from state s the greater the value of $Q(s, a)$. The actor's decision rule defines the precise relationship between attractions and action probabilities:

Definition 8. A **decision rule** is a function $\mathcal{D}$ that transforms an attraction function $Q^{(t)}$ into a policy $\pi^{(t)}$.

The following examples of decision rules are from [Camerer and Ho, 1999]:

[8] $U((s_0, s_1, ...)) = U(s_0) + \delta \sum_{t=1}^{\infty} \gamma^t U(s_t)$, where γ is the same as it is in exponential discounting, and $\delta \in [0, 1)$

Example 1. Argmax Decision Rule. The actor chooses uniformly at random among the actions with maximum attractiveness.

$$\pi^{(t)}(s,a) = \begin{cases} \frac{1}{|\arg\max_{a' \in A} Q^{(t)}(s,a')|}, & a \in arg\max_{a' \in A} Q^{(t)}(s,a') \\ 0, & otherwise \end{cases} \tag{2}$$

Example 2. Softmax Decision Rule. The actor chooses an action with probability exponential in its attractiveness.

$$\pi^{(t)}(s,a) = \frac{\exp\{\tau^{(t)} Q^{(t)}(s,a)\}}{\sum_{a' \in A} \exp\{\tau^{(t)} Q^{(t)}(s,a')\}} \tag{3}$$

Notice that the only purpose of the denominator is to normalize the probabilities.

$\tau^{(t)}$ is a non-negative parameter that can be interpreted as the confidence of the actor in his attractions at time t. Two special cases illustrate this point: 1) as $\tau^{(t)} \to \infty$, softmax approaches argmax, and 2) when $\tau^{(t)} = 0$, softmax produces policies that choose actions uniformly at random, completely ignoring attractions.

Example 3. Power Decision Rule.

$$\pi^{(t)}(s,a) = \frac{Q^{(t)}(s,a)^{\tau^{(t)}}}{\sum_{a' \in A} Q^{(t)}(s,a')^{\tau^{(t)}}} \tag{4}$$

Everything said about $\tau^{(t)}$ in the softmax decision rule is true in the the power decision rule.

Definition 9. A **learning rule** is a function $\mathcal{L}$ that takes in the interaction history $H^{(t)}$ and the actor's utility function U and produces attractions Q.

Note that because the learning rule takes in the interaction history at any time, including time 0, it implicitly defines the initial attractions as a function of the actor's utility function: $\mathcal{L}(H^{(0)}, U)$ gives the actor's initial attractions.

Example 4. Q Learning.

Algorithm 1 implements the Q Learning rule, as seen in [Sutton and Barto 1998], which generates Q from $\mathcal{H}^{(T)}$. The function has two parameters. One is a **learning rate schedule**, which is a sequence $\{\alpha_t\}$ of numbers $\alpha_t \in [0,1]\ \forall t$. The other is a **time discount factor** $\gamma \in [0,1)$, used for exponential discounting. At each timestep, the algorithm estimates, based on current attractions, the time-discounted sum of utilities $Y = \mathbb{E}\left[U(s_{t+1}) + \gamma U(s_{t+2}) + \gamma^2 U(s_{t+3}) + ...\right]$ that the actor can earn by choosing actions optimally from s_{t+1}, and then updates $Q_{s_t a_t} \leftarrow (1-\alpha_t) Q_{s_t a_t} + \alpha_t Y$. As one can see from the update rule, an interpretation of α_t is that it is the sensitivity of attractions to new experiences.

Algorithm 1 Q Learning

Initialize $Q(s,a) \leftarrow \frac{1}{(1-\gamma)|S|}\sum_{s' \in S} U(s')$ for all $s, a \in S \times A$
for all $t \in (0, 1, ..., T)$ **do**
  Get (s_t, a_t, s_{t+1}) from $\mathcal{H}^{(T)}$
  $Q(s_t, a_t) \leftarrow (1-\alpha_t)Q(s_t, a_t) + \alpha_t \left[U(s_{t+1}) + \gamma \max_{a' \in A} Q(s_{t+1}, a')\right]$
return Q

Example 5. SARSA Learning. Algorithm 2 implements the SARSA learning rule, also seen in [Sutton and Barto, 1998], which is exactly the same as Q Learning, except that when estimating the future utility, it uses the actually chosen a_{t+1} instead of the maximizing action. This difference is important for methods that I will discuss later.

Algorithm 2 SARSA Learning

Initialize $Q(s,a) \leftarrow \frac{1}{(1-\gamma)|S|}\sum_{s' \in S} U(s')$ for all $s, a \in S \times A$
for all $t \in (0, 1, ..., T-1)$ **do**
  Get $(s_t, a_t, s_{t+1}, a_{t+1})$ from $\mathcal{H}^{(T)}$
  $Q(s_t, a_t) \leftarrow (1-\alpha_t)Q(s_t, a_t) + \alpha_t \left[U(s_{t+1}) + \gamma Q(s_{t+1}, a_{t+1})\right]$
return Q

A Note on Notation

Since there are finitely many states and actions, it is convenient to represent the functions whose inputs are states and actions as vectors and matrices. For examples $U_s := U(s)$, $\pi_{sa} = \pi(s,a)$, and $T_{sas'} := T(s,a,s')$. Additionally, I will sometimes 'curry' the functions: $\pi_s = \pi(s,.)$ gives a vector representation of the probability distribution over actions to take from state s, for example. This notation allows the use of linear algebra operations like the dot product $\cdot$ and the vector transpose $\top$.

3.3 The Observer

The observer's goal is to infer the utility function of the actor, based on the interaction history, knowing the learning and decision rules of the actor and potentially also the reward function of the environment.

Without loss of generality[9], the observer uses a featured representation of U, as seen in [Abbeel and Ng, 2004]. For a fixed collection of feature functions $\phi_1, \phi_2, ..., \phi_n$,

[9]There are finitely many states, so the observer could infer the utility function directly by using the features $\phi_i(s) = 1$ if $s = i$ and $\phi_i(s) = 0$ if $s \neq i$, for all $i \in S$.

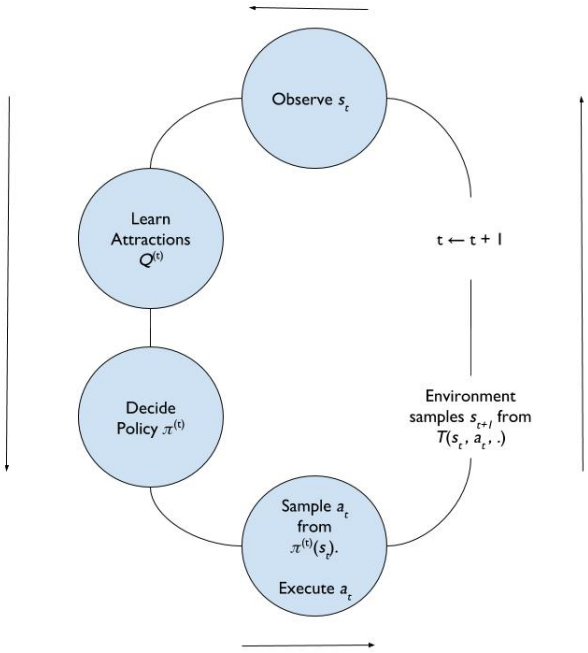

Figure 1: A diagram of the actor-environment interface.

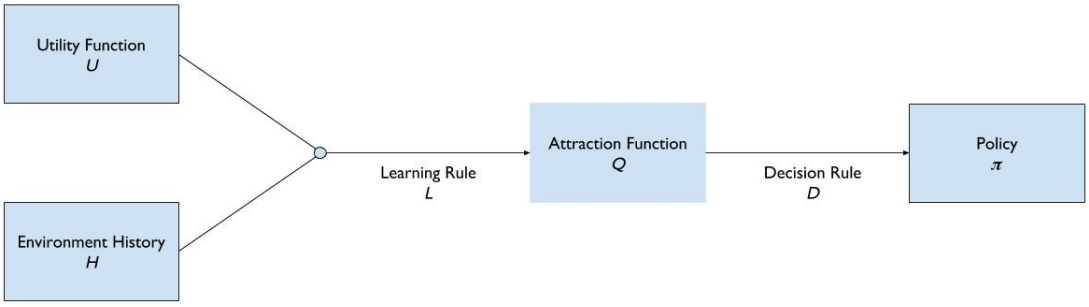

Figure 2: A diagram of an actor.

where $\phi_i : S \to \mathbb{R}$ for all i, and feature weights $\theta_1, \theta_2, ..., \theta_n$

$$U_s = \sum_i \theta_i \phi_i(s) = \theta \cdot \phi(s) \tag{5}$$

Equivalently, if Φ is defined to be the matrix for which $\Phi_{si} = \phi_i(s)$, then

$$U = \Phi\theta \tag{6}$$

The features allow the observer to focus on information that is relevant to her. Perhaps she doesn't want a full model of the the actor's utility function in this one environment; perhaps she only cares to know whether the actor prefers the highway or backroads (which may be of interest to a self-driving car), or how the actor trades off selfishness and altruism.

Feature functions also allow the observer to incorporate any information that she happens to have about the environment's reward function; an estimate of the environment's reward function would make a great feature, because most actors probably react to the environment in some way.

Like in any statistical problem, conclusions generated from a small amount of data or, in this case, even a large amount of data from just one environment should be taken with a grain of salt. However, the features allow the observer to do the best she can with the information she has.

3.4 Statement of the Problem

The methods presented here solve the observer's problem, which is stated below.

Given: The sets S and A describing the environment, the actor's learning rule $\mathcal{L}$ and decision rule $\mathcal{D}$, and fixed features Φ,

Determine: Feature weights θ such that $\Phi\theta = U$, where U is the vector representation of the utility function that the actor is optimizing.

Assuming that: The actor acts based on his utility function, learning rule, and decision rule.

4 Future-Belief Learning Rules

The solutions in the next sections are for two special cases of the problem. To nail down what these cases are, it is necessary to define what I call a future-belief learning rule.

Definition 10. An attraction value $Q_{sa}^{(t)}$ is said to be **rationalized by future-beliefs** if there exists a state-indexed vector $B_{sa}^{(t)}$, which can be determined from $\mathcal{H}^{(t-1)}$ (without U), where $B_{sas'}^{(t)} \in [0, B_{\max}]$ (where $B_{\max} > 0$) for every state s', such that

$$Q_{sa}^{(t)} = (B_{sa}^{(t)})^{\top} U \tag{7}$$

where $\top$ denotes vector transposition. The full tensor $B^{(t)}$ contains the actor's **future-beliefs** at time t.

Definition 11. A learning rule is a **future-belief learning rule** if all attractions that it learns are rationalized by future-beliefs.

How should one interpret future-beliefs? Loosely speaking, they are beliefs that the actor holds about what he will experience in the future, conditional on taking a particular action from a particular state. To be slightly more precise, $B_{sas'}$ gives the actor's expectation of the time-discounted number of times that he will ever arrive at state s' after taking action a from state s, assuming that he makes decisions roughly as he has in the past. For example, suppose that the actor knows for certain that after taking action a from state s, he will arrive at $s' \neq s$; and then, after choosing another action, he will arrive at s; and then at s'; and then at some new state s_T; and then the MDP will terminate. In this case, if he uses exponential discounting, his future-belief $B_{sas'}$ is $1 \cdot \gamma^0 + 0 \cdot \gamma^1 + 1 \cdot \gamma^2 + 0 \cdot \gamma^3 = 1 + \gamma^2$.

Future-belief learning should not be confused with model-based reinforcement learning. In general, model-based learning rules are not future-belief learning rules, and future-belief learning rules are not model-based learning rules.

Example 6. Recall the SARSA learning rule in Example 5. Algorithm 3 learns the future-beliefs that rationalize the attractions that SARSA learns, and thus, SARSA is a future-belief learning rule.

Algorithm 3 SARSA Future-Belief Learning

Initialize $B_{sas'} \leftarrow \frac{1}{(1-\gamma)|S|}$ for all $s, a, s' \in S \times A \times S$
for all $t \in (0, 1, ..., T-1)$ **do**
  Get $(s_t, a_t, s_{t+1}, a_{t+1})$ from $\mathcal{H}^{(T)}$
  $B_{s_t,a_t} \leftarrow (1-\alpha_t) B_{s_t,a_t} + \alpha_t \left[I(s_{t+1}) + \gamma B_{s_{t+1},a_{t+1}} \right]$
    ▷ $I(s)$ is the vector v for which $v_s = 1$ and $v_i = 0 \; \forall i \neq s$
return B

Theorem 1. SARSA (see Example 5) is a future-belief learning rule.

Proof. First,

$$U \cdot B_{sa}^{(0)} = \frac{1}{(1-\gamma)|S|} \sum_{s'} U_{s'} = Q_{sa}^{(0)} \tag{8}$$

for all $s, a \in S \times A$, so SARSA's initial attractions are rationalized by future-beliefs. Now, suppose that $Q_{sa}^{(t)} = U \cdot B_{sa}^{(t)}$. At time $t+1$, Q_{s_t,a_t} and B_{s_t,a_t} are updated upon the actor's observation of $(s_t, a_t, s_{t+1}, a_{t+1})$ in the ways defined in Algorithms 2 and 3, respectively. For all $s, a \neq (s_t, a_t)$, nothing changes: $Q^{(t+1)}(s,a) = Q_{sa}^{(t)} = U \cdot B_{sa}^{(t)} = U \cdot B_{sa}^{(t+1)}$. On the other hand, for $(s,a) = (s_t, a_t)$:

$$\begin{aligned}
U \cdot B_{sa}^{(t+1)} &= U \cdot \left[(1-\alpha_t)B_{sa}^{(t)} + \alpha_t[I(s_{t+1}) + \gamma B_{s_{t+1},a_{t+1}}^{(t)}]\right] \\
&= (1-\alpha_t)U \cdot B_{sa}^{(t)} + \alpha_t \left[U \cdot I(s_{t+1}) + \gamma U \cdot B_{s_{t+1},a_{t+1}}^{(t)}\right] \\
&= (1-\alpha_t)Q_{sa}^{(t)} + \alpha_t \left[U_{s_{t+1}} + \gamma Q_{s_{t+1},a_{t+1}}^{(t)}\right] \\
&= Q_{sa}^{(t+1)}
\end{aligned}$$

So for any time t and any $s, a \in S \times A$, $Q_{sa}^{(t)} = U \cdot B_{sa}^{(t)}$. It is clear that $B_{sa}^{(t)} \geq 0$. The only thing left to prove is that the future-beliefs are bounded: that there exists $B_{\max}$ such that $B_{sas'}^{(t)} \leq B_{\max}$. I show, using the same proof strategy as before, that this is satisfied for $B_{\max} = \frac{1}{1-\gamma}$.

Clearly $B_{sas'}^{(0)} = \frac{1}{(1-\gamma)|S|} \leq \frac{1}{1-\gamma}$. If $B_{sas'}^{(t)} \leq \frac{1}{1-\gamma}$ $\forall s, a, s' \in S \times A \times S$, then

$$\begin{aligned}
B_{s_t,a_t,s'}^{(t+1)} &= (1-\alpha_t)B_{s_t,a_t}^{(t)} + \alpha_t \left[1_{s'=s_{t+1}} + \gamma B_{s_{t+1},a_{t+1}}^{(t)}\right] \\
&\leq (1-\alpha_t)\frac{1}{1-\gamma} + \alpha_t \left[1 + \frac{\gamma}{1-\gamma}\right] \\
&= \frac{1-\alpha_t}{1-\gamma} + \frac{\alpha_t(1-\gamma)}{1-\gamma} + \frac{\gamma\alpha_t}{1-\gamma} \\
&= \frac{1}{1-\gamma}
\end{aligned}$$

□

5 Linear Program Solution for Argmax Decisions

In this section, the actor uses a future-belief learning rule and the argmax decision rule (see Example 1). Read the next section to see an alternative special case in which the actor uses a future-belief learning rule and the *softmax* decision rule.

Because the actor uses argmax, each decision that he makes imposes a constraint on his attractions and, transitively, on his utility function, from the observer's perspective. If the actor chooses action a_t at time t, then the observer knows immediately that

there is no action to which the actor was more attracted at time t than a_t. Moreover, because the actor is a future-belief learner, these constraints are linear in the actor's utility vector. This is why it is natural to take inspiration from [Ng and Russell, 2000] and formulate the problem as a linear program.

5.1 Constraints

As mentioned above, if an actor chooses action a_t at time t, then action a_t has a weakly greater attraction at time t than every other action available to the actor. Mathematically,

$$Q^{(t)}_{s_t,a_t} \geq Q^{(t)}_{s_t,a} \tag{9}$$

for all times $0 \leq t \leq T$ and actions $a \in A \setminus a_t$. So, at time T, there are $(|A| - 1)T$ constraints of this kind.

Writing this in terms of the utility parameters θ, using Equations 6 and 7,

$$(B^{(t)}_{s_t,a_t})^\top \Phi\theta \geq (B^{(t)}_{s_t,a})^\top \Phi\theta \tag{10}$$

or, rearranging terms,

$$\left[(B^{(t)}_{s_t,a_t})^\top - (B^{(t)}_{s_t,a})^\top\right] \Phi\theta \geq 0 \tag{11}$$

Under the argmax decision rule, attractions are unique only up to a linear transformation with positive slope. That is, if one were to construct attractions Q' from attractions Q as such:

$$Q' = mQ + b \tag{12}$$

for $m > 0$, the argmax decision rule would produce the same decision distributions under Q' as it would under Q.

Thus it is without loss of generality that I constrain

$$0 \leq \theta_i \leq 1 \tag{13}$$

for all features $1 \leq i \leq n$. It is convenient that we can use these constraints WLOG, because without them, it is possible for the linear program to be unbounded. There are $2n$ constraints of this type, for a total of $(|A| - 1)T + 2n$ constraints.

5.2 Objective Function

In general, there are many θs that satisfy the constraints. Some of them are not useful. In particular, a choice of θ that produces the same utility for every state (e.g., $U = \mathbf{0}$)

always satisfies the constraints. We would like to have an objective function that privileges utility functions that explain the actor's decisions better than other utility functions. I propose using the following objective function:

$$\sum_{t=0}^{T}\left[Q_{s_t,a_t}-\frac{1}{|A|}\sum_{a\in A}Q_{s_t,a}\right] \tag{14}$$

$$=\sum_{t=0}^{T}\left[(B^{(t)}_{s_t,a_t})^\top\Phi\theta-\frac{1}{|A|}\sum_{a\in A}(B^{(t)}_{s_t,a})^\top\Phi\theta\right]$$

$$=\left(\sum_{t=0}^{T}\left[(B^{(t)}_{s_t,a_t})^\top-\frac{1}{|A|}\sum_{a\in A}(B^{(t)}_{s_t,a})^\top\right]\Phi\right)\theta \tag{15}$$

Equation 14 can be interpreted as the difference between the attraction of the chosen action a_t and the average attraction of other actions, all summed over time. This objective function is always nonnegative for values of θ that satisfy the constraints, and it is 0 for the trivial solutions that produce constant utility functions. Thus, the objective function never privileges a trivial solution over a nontrivial one.

Intuitively, this objective function chooses the θ that creates the greatest average gap between the attractions to the decisions chosen and the attractions to decisions not chosen.

6 MAP Solution for Softmax Decisions

In this section, the actor uses a future-belief learning rule and the softmax decision rule, as presented in Example 2, which does not necessarily choose an action with the greatest attraction. There are many reasons why softmax might be a more practical decision rule to consider than argmax:

- The actor might make mistakes.
- The actor may sometimes choose a less attractive action for the purpose of exploring the environment.[10]
- The actor might enjoy variety. This is especially relevant in the domain of market research.
- In practice, the observer may be slightly wrong about the actor's learning rule, in which case the room for error that a non-maximizing decision rule allows is essential.

[10]The parameter, τ, can be adjusted to achieve different levels of exploration.

In this setting, it is useful to frame the problem as one of maximum a posteriori (MAP) estimation: the observer has a prior p over the feature weights θ that model the actor's utility function and infers that the utility function is the one that maximizes the posterior distribution generated from p and $\mathcal{H}$.

6.1 Additional Notation

Let $\pi^{(t)}(\theta) := \mathcal{D}(\mathcal{L}(\Phi\theta, \mathcal{H}^{(t-1)}))$, the policy that the actor would learn based on $\mathcal{H}^{(t)}$ if his utility function were the vector $\Phi\theta$. Let $s_{j:k} = (s_j, s_{j+1}, ..., s_k)$, and define the analogous notation for actions.

6.2 The Posterior

The posterior is

$$
\begin{aligned}
L^{(T)}(\theta) &:= \Pr(U = \Phi\theta \mid \mathcal{H}^{(T)}) \\
&= c_1 p(\theta) \Pr(\mathcal{H}^{(T)} \mid U = \Phi\theta) && \text{Bayes' Rule} \\
&= c_1 p(\theta) \Pr(s_{0:T}, a_{0:T} \mid U = \Phi\theta) && \text{Def of } \mathcal{H}^{(T)} \\
&= c_1 p(\theta) \prod_{t=0}^{T} \Pr(s_t, a_t \mid U = \Phi\theta, s_{0:t-1}, a_{0:t-1}) && \text{Def of conditional prob} \\
&= c_1 p(\theta) \prod_{t=0}^{T} T(s_{t-1}, a_{t-1}, s_t) \Pr(a_t | s_t, U = \Phi\theta, \mathcal{H}^{(t-1)}) && (16) \\
&= c p(\theta) \prod_{t=0}^{T} \Pr(a_t \mid s_t, U = \Phi\theta, \mathcal{H}^{(t-1)}) && (17) \\
&= c p(\theta) \prod_{t=0}^{T} \pi^{(t)}(\theta)_{s_t, a_t} && (18)
\end{aligned}
$$

where:

- In the Bayes' Rule step, c_1 is the normalizing constant.
- (16) follows from splitting the factors inside the product into $\Pr(s_t \mid U = \Phi\theta, s_{0:t-1}, a_{0:t-1}) \Pr(a_t \mid s_t, U = \Phi\theta, s_{0:t-1}, a_{0:t-1})$ and then noticing that the former factor is $T(s_{t-1}, a_{t-1}, s_t)$; the other variables are irrelevant. For simplicity in the notation, define $T(s_{-1}, a_{-1}, s_0)$ to be the (potentially deterministic) probability distribution over initial states s_0.
- (17) follows from noticing that $T(s_{t-1}, a_{t-1}, s_t)$ does not depend on θ and pulling it out as a constant: $c := c_1 \prod_{t=1}^{T} T(s_{t-1}, a_{t-1}, s_t)$.

- (18) follows from the definition of $\pi^{(t)}(\theta)$.

As is often the case, it is easier to work with the negative logarithm of the posterior $l(\theta) := -\log L(\theta)$ than the posterior itself. Using Equation 18 and properties of logarithms,

$$l(\theta) = -\log c - \log p(\theta) - \sum_{t=0}^{T} \log \pi^{(t)}(\theta)_{s_t,a_t} \tag{19}$$

Finding $\arg\min_\theta l(\theta)$ is an ill-posed problem; there may be no solution, since there may always be a way to change θ to decrease l. To fix this, I regularize $l(\theta)$ by choosing a regularization constant $r > 0$ and adding $\frac{r}{2}||\theta||_2^2$. The final posterior is:

$$l_r(\theta) = \frac{r}{2}||\theta||_2^2 - \log c - \log p(\theta) - \sum_{t=0}^{T} \log \pi^{(t)}(\theta)_{s_t,a_t} \tag{20}$$

6.3 Gradient Descent

The approach to minimizing l_r is to attempt to use gradient descent with backtracking line search [Gordon and Tibshirani, 2012], which is outlined in Algorithm 4.

Algorithm 4 General Gradient Descent

Require: backtracking line search parameters $\beta \in (0, 1)$
Require: stop condition parameter $\epsilon > 0$
  $\theta \leftarrow [0, 0, ..., 0]$ (length N)
  $\mathbf{g} \leftarrow$ computePosteriorGradient$(\mathcal{H}, \mathcal{L}, \mathcal{D}, \theta)$
  while $||\mathbf{g}||_2 > \epsilon$ **do**
    $\sigma \leftarrow 1$ ▷ step size
    while $l_r(\theta - \sigma g) > l_r(\theta) - \frac{\sigma}{2}||g||_2^2$ **do**
      $\sigma \leftarrow \beta\sigma$ ▷ Decrease step size σ until the step will not increase l_r by too much
    $\theta \leftarrow \theta - \sigma\mathbf{g}$ ▷ Update θ
    $\mathbf{g} \leftarrow$ computePosteriorGradient$(\mathcal{H}, \mathcal{L}, \mathcal{D}, \theta)$
  return θ

The hairy part of the process, which Algorithm 4 abstracts out, is to compute the gradient of l_r, which is:

$$\bigtriangledown l_r(\theta) = r\theta - \frac{\bigtriangledown p(\theta)}{p(\theta)} - \sum_{t=0}^{T} \frac{\bigtriangledown \pi^{(t)}(\theta)_{s_t,a_t}}{\pi^{(t)}(\theta)_{s_t,a_t}} \tag{21}$$

Since the decision rule is softmax and the attractions are rationalized by future-beliefs,

$$\pi^{(t)}(\theta)_{sa} = \frac{N_{sa}^{(t)}(\theta)}{D_s^{(t)}(\theta)} \tag{22}$$

where the (N)umerator is:

$$N_{sa}^{(t)}(\theta) := \exp\{\tau^{(t)}(B_{sa}^{(t)})^\top \Phi\theta\} \tag{23}$$

and the (D)enominator is:

$$D_s^{(t)}(\theta) := \sum_{a' \in A} N_{sa'}^{(t)}(\theta) \tag{24}$$

Note on Notation. There is a lot of dense math in what follows, so it is helpful to introduce some shortcuts in the notation. From here on out, (t) subscripts and superscripts are dropped where they are implied. Superscript letters after functions are used to indicate partial derivatives. For example,

$$\pi(\theta)_{sa}^i := \frac{\partial}{\partial \theta_i} \pi(\theta)_{sa} \tag{25}$$

In addition, it helps to define

$$f_{sa} := B_{sa}^\top \Phi \tag{26}$$

so that $N_{sa}(\theta)$ can be written as $\exp\{\tau f_{sa} \cdot \theta\}$. Intuitively, where B_{sa} is a vector of future-beliefs about states, f_{sa} is a vector of future-beliefs about features. While $B_{sas'}$ is the time-discounted expected number of times that the actor will reach state s' after taking action a from state s, f_{sai} is the actor's time-discounted expected sum, over future states that he expects to reach, of feature i. This is reminiscent of the "feature expectations" in [Abbeel and Ng, 2004].

Now, on to the dense math. To use a form of Algorithm 4, it is necessary to have a formula for $\frac{\bigtriangledown \pi(\theta)_{sa}}{\pi(\theta)_{sa}}$. First, the partial derivatives of the numerator and denominator:

$$N_{sa}(\theta)^i = \tau f_{sai} N_{sa}(\theta) \tag{27}$$

so

$$D_s(\theta)^i = \sum_{a' \in A} N_{sa'}(\theta)^i = \tau \sum_{a' \in A} f_{sa'i} N_{sa'}(\theta) \tag{28}$$

Now,

$$\pi(\theta)^i_{sa} := \left[\frac{N_{sa}(\theta)}{D_s(\theta)}\right]^i$$

$$= \frac{N_{sa}(\theta)^i D_s(\theta) - D_s(\theta)^i N_{sa}(\theta)}{D_s(\theta)^2} \tag{29}$$

$$= \frac{\tau f_{sai} N_{sa}(\theta)}{D_s(\theta)} - \frac{\tau \sum_{a' \in A} f_{sa'i} N_{sa'}(\theta)}{D_s(\theta)} \frac{N_{sa}(\theta)}{D_s(\theta)} \tag{30}$$

$$= \tau f_{sai} \pi(\theta)_{sa} - \tau \left[\sum_{a' \in A} f_{sa'i} \pi(\theta)_{sa'}\right] \pi(\theta)_{sa} \tag{31}$$

$$= \tau \pi(\theta)_{sa} \left(f_{sai} - \sum_{a' \in A} f_{sa'i} \pi(\theta)_{sa'}\right) \tag{32}$$

$$= \tau \pi(\theta)_{sa} \left(f_{sai} - \mathbb{E}_{a' \sim \pi(\theta)_s}[f_{sa'i}]\right) \tag{33}$$

where:

- (29) follows from the quotient rule of derivatives.
- (30) results from plugging in Equations 27 and 28 and splitting the fraction.
- (31) results from rewriting each $\frac{N_{sa}(\theta)}{D_s(\theta)}$ as $\pi(\theta)_{sa}$.
- (32) results from factoring.
- (33) rewrites (32), noticing that the sum is an expectation, over actions sampled from $\pi(\theta)_s$.

Thus,

$$\frac{\bigtriangledown \pi(\theta)_{sa}}{\pi(\theta)_{sa}} = \tau \left(f_{sa} - \mathbb{E}_{a' \sim \pi(\theta)_s}[f_{sa'}]\right) \tag{34}$$

Now we can write Algorithm 5, a more detailed version of Algorithm 4, which specifies the gradient computation and also exploits the future-beliefs to avoid repeated computation.

It would be ideal if Algorithm 5 were guaranteed to converge to the parameters θ that minimize $l_r(\theta)$. Gradient descent is guaranteed [Gordon and Tibshirani, 2012] to converge to a global minimum of l_r if l_r:

1. is a convex function of θ
2. has a Lipschitz-continuous gradient

The next two sections prove each respectively.

Algorithm 5 Inference with Future-Belief Learning and Softmax Decisions

Require: line search parameter $\beta \in (0, 1)$
Require: stop condition parameter $\epsilon > 0$
  $\theta \leftarrow [0, 0, ..., 0]$ (length n)
  for all $0 \leq t \leq T$ **do**
    Compute future-beliefs $B^{(t)}$ from $\mathcal{H}^{(t)}$, using the actor's learning rule
    for all $s, a \in S \times A$ **do**
      $f_{sa}^{(t)} \leftarrow (B_{sa}^{(t)})^\top \Phi$ ▷ Compute and store feature future-beliefs
  $\mathbf{g} \leftarrow$ computePosteriorGradient(f, θ)
  while $||\mathbf{g}||_2 > \epsilon$ **do**
    $\sigma \leftarrow 1$ ▷ step size
    while $l_r(\theta - \sigma g) > l_r(\theta) - \frac{\sigma}{2}||g||_2^2$ **do**
      $\sigma \leftarrow \beta\sigma$ ▷ Decrease step size σ until the step will not increase l_r by too much
    $\theta \leftarrow \theta - \sigma\mathbf{g}$ ▷ Update θ, based on the computed gradient $\mathbf{g}$ and line search parameters.
    $\sigma \leftarrow \beta\sigma$ ▷ Scale the step size down.
    $\mathbf{g} \leftarrow$ computePosteriorGradient(f, θ)
  return θ

  function COMPUTEPOSTERIORGRADIENT(f, θ)
    $g \leftarrow r\theta - \frac{\bigtriangledown p(\theta)}{p(\theta)}$ ▷ gradient of regularization term and log prior
    for all $t \in 0, 1, ..., T$ **do**
      $g \leftarrow g - \tau^{(t)} \left(f_{s_t, a_t}^{(t)} - \sum_{a' \in A} \pi^{(t)}(\theta)_{s_t, a'} f_{s_t, a'}^{(t)} \right)$
    return g

6.4 Convexity

Theorem 2. Suppose that the prior p is a log-concave[11] function of θ. Then l_r is a convex function of θ.

Proof. Since $||.||_2$ is convex, and $f(x) = x^2$ is convex and non-decreasing, $||.||_2^2$ is convex. Since p is log-concave, $-\log p$ is convex. The positive-weighted sum of convex functions is convex, so if $-\log \pi^{(t)}(\theta)_{s_t,a_t}$ is convex at all times t, then l_r is convex. We now prove that $-\log \pi^{(t)}(\theta)_{s_t,a_t}$ is convex by proving that its Hessian is positive-semidefinite.

Based on Equation 32, and dropping time notation as before,

$$[-\log \pi(\theta)_{sa}]^i = -\frac{\pi(\theta)^i_{sa}}{\pi(\theta)_{sa}} = -\tau \left(f_{sai} - \sum_{a'} \pi(\theta)_{sa'} f_{sa'i} \right) \tag{35}$$

Taking the partial derivative with respect to θ_j, for $j \neq i$:

$$\begin{aligned}
&\left[-\tau \left(f_{sai} - \sum_{a'} \pi(\theta)_{sa'} f_{sa'i} \right) \right]^j \\
&= \tau \sum_{a'} \pi(\theta)^j_{sa'} f_{sa'i} \\
&= \tau^2 \sum_{a'} \pi(\theta)_{sa'} \left(f_{sa'j} - \mathbb{E}_{a'' \sim \pi(\theta)_s}[f_{sa''j}] \right) f_{sa'i} && (36) \\
&= \tau^2 \left(\sum_{a'} \pi(\theta)_{sa'} f_{sa'j} f_{sa'i} - \sum_{a'} \pi(\theta)_{sa'} \mathbb{E}_{a'' \sim \pi(\theta)_s}[f_{sa''j}] f_{sa'i} \right) && (37) \\
&= \tau^2 \left(\sum_{a'} \pi(\theta)_{sa'} f_{sa'j} f_{sa'i} - \mathbb{E}_{a' \sim \pi(\theta)_s}[f_{sa'j}] \sum_{a'} \pi(\theta)_{sa'} f_{sa'i} \right) && (38) \\
&= \tau^2 \left(\mathbb{E}_{a' \sim \pi(\theta)_s} \left[f_{sa'i} f_{sa'j} \right] - \mathbb{E}_{a' \sim \pi(\theta)_s} \left[f_{sa'i} \right] \mathbb{E}_{a' \sim \pi(\theta)_s} \left[f_{sa'j} \right] \right) && (39) \\
&= \tau^2 \mathrm{Cov}_{a' \sim \pi(\theta)_s} \left[f_{sa'i}, f_{sa'j} \right] && (40)
\end{aligned}$$

where:

- (36) follows from plugging in Equation 33 for $\pi(\theta)^j_{sa'}$.
- (37) results from splitting the sum.

[11] A function is log-concave if its logarithm is concave. This is not a very restrictive assumption. Many common distributions, such as uniform, normal, and exponential, are log-concave.

- (38) results from factoring $\mathbb{E}_{a'' \sim \pi(\theta)_s}[f_{sa''j}]$ out of the sum on the right (and then simplifying notation by creating two separate domains of a').
- (39) results from replacing the sums with their equivalent expectations.
- (40) follows from the fact that for any random variables X and Y, $\text{Cov}[X, Y] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$.

One can show through similar steps that

$$\frac{\partial^2(-\log \pi(\theta)_{sa})}{\partial \theta_i^2} = \tau^2 \text{Var}_{a' \sim \pi(\theta)_s}[f_{sa'i}] \tag{41}$$

Thus, the Hessian of $-\log \pi(\theta)_{sa}$ is $\tau^2 C$, where C is the covariance matrix for which $C_{ij} := \text{Cov}_{a' \sim \pi(\theta)_s}[f_{sa'i}, f_{sa'j}]$:

$$\bigtriangledown^2(-\log \pi(\theta)_{sa}) = \tau^2 C \tag{42}$$

Since covariance matrices are positive semidefinite, the Hessian is positive semidefinite. This completes the proof.

□

6.5 Lipschitz-Continuous Gradient

Definition 12. A function $f : \mathbb{R}^n \to \mathbb{R}$ is **Lipschitz-continuous** if there is a constant, called the **Lipschitz constant**, $L > 0$ for which for any $u, v \in \mathbb{R}^n$,

$$||f(u) - f(v)||_2 \leq L||u - v||_2 \tag{43}$$

Theorem 3. Suppose that $\bigtriangledown(-\log p)$ is Lipschitz-continuous.[12] Then $\bigtriangledown l_r$ is Lipschitz-continuous.

Proof. The gradient of the regularization term $\bigtriangledown \frac{r}{2}||\theta||_2^2 = r\theta$ is clearly a Lipschitz-continuous function of θ, with Lipschitz constant r. The sum of two Lipschitz-continuous functions is Lipschitz-continuous (and the sum of their Lipschitz constant works as the Lipschitz constant), so, since the gradient of the prior term is Lipschitz-continuous by assumption, it is sufficient to prove that $\bigtriangledown(-\log \pi(\theta_{sa}))$ is

[12] This, too, is not very restrictive of priors. The uniform, normal, and exponential distributions all satisfy this assumption.

Lipschitz-continuous. Because $-\log \pi(\theta)_{sa}$ is convex and twice-differentiable in θ, it is equivalent too prove that there exists $L > 0$ for which

$$\bigtriangledown^2(-\log \pi(\theta)_{sa}) = \tau^2 C \preceq LI \tag{44}$$

where I is the identity matrix. This is to say that $\tau^2 C - LI$ is negative-semidefinite. We show that this is true for $L = \tau^2 n C_{\max}$, where n is the number of feature functions, and $C_{\max}$ is the maximum element of C.

Take an arbitrary vector v of length n. We want to show that $v^\top(\tau^2 C - \tau^2 n C_{\max} I)v \leq 0$.

$$\begin{aligned}
& v^\top(\tau^2 C - \tau^2 n C_{\max} I)v \\
&= \tau^2 \left(v^\top C v - n C_{\max} ||v||_2^2 \right) \\
&= \tau^2 \left(\left(\frac{v}{||v||_1} \right)^\top C \left(\frac{v}{||v||_1} \right) ||v||_1^2 - n C_{\max} ||v||_2^2 \right) \\
&\leq \tau^2 \left(C_{\max} ||v||_1^2 - n C_{\max} ||v||_2^2 \right) && (45) \\
&= \tau^2 C_{\max} \left(||v||_1^2 - n ||v||_2^2 \right) && (46)
\end{aligned}$$

where (45) follows from the observation that $(\frac{v}{||v||_1})^\top C(\frac{v}{||v||_1})$ is a weighted average of entries of C, since the sum of the entries of $\frac{v}{||v||_1}$ is 1.

The term $\tau^2 C_{\max}$ in Equation 46 is non-negative (since C contains variances, which are non-negative), so it is sufficient to prove that $||v||_1^2 - n||v||_2^2$ is non-positive. For this, we use the Cauchy-Schwarz inequality, that for any vectors u and v, $|u \cdot v| \leq ||u||_2 ||v||_2$. To apply this, let u be an n-length vector of all ones and v be as it is defined in Equation 46. Then $|u \cdot v| = ||v||_1$, and $||u||_2 = \sqrt{n}$, so it follows from the Cauchy-Schwarz inequality that $||v||_1 \leq \sqrt{n}||v||_2$, so $||v||_1^2 \leq n||v||_2^2$. This completes the proof that $\bigtriangledown l_r$ is Lipschitz-continuous. □

7 Future Work

I view the work in this thesis as a first step from standard inverse reinforcement learning toward practical application. There are both empirical and theoretical avenues to improvement.

7.1 Empirical Work

The most immediate work to be done is to run experiments to demonstrate the extent to which the solutions presented in the previous sections are effective. The following

experiment, inspired by a Turing test, would be particularly interesting:

1. Human participants, motivated by a specified payoff structure, play the role of actor in $m \geq 2$ different MDPs. Although the MDPs are different, they share some common features.

2. Some artificial agents also (separately) play the role of actor in the same MDPs. They are motivated by the same payoff structure as the human actors.

3. A separate set of human participants play the role of observer in the same MDPs. Each observer is paired with a single actor. The observer observes the actions of her actor and, after each of his actions, predicts his next action. The observers do not know the payoff structure that motivates the actors.

4. Artificial agents, some equipped with the linear program solution and others equipped with the MAP solution, do the same thing as the human observers.

The purpose of this experiment is to determine whether humans or my artificial agents are better at predicting the future behavior of the actors, or that there is no significant difference. It could be run efficiently on Amazon Mechanical Turk.

I would also be interested to know which learning and decision rules make the best predictions of human behavior with this method. It is very difficult in behavioral research to determine one of the actor's components–utility function, learning rule, decision rule–without assuming values of the others. This kind of experiment would be interesting because it would find explanatory learning and decision rules without making assumptions about the actor's utility function.

7.2 Theoretical Work

On the theoretical side, the smallest step in the direction of progress is to explore other special cases of the problem presented here. Is the gradient method still guaranteed to converge if the actor uses the power decision rule (see Example 3) instead of the softmax decision rule? That might be helpful in some applications, since under the power decision rule, decisions are unaffected by a positive linear transformation of the attractions, which means that one can constrain the utility parameters θ to be between 0 and 1, like in the linear program formulation for argmax decisions. It would be even better, of course, to identify a more general set of conditions on decision rules under which gradient descent is guaranteed to converge. One might also search for solutions to this problem that have theoretical guarantees even when the actor's learning rule is not a future-belief learning rule, but it should first be confirmed empirically that future-belief learning rules fit data worse than other learning

rules.[13]

The next steps involve relaxing some of the assumptions about the knowledge of the observer. Here, the observer was assumed to know the actor's 1) learning rule, which includes his 2) initial attractions as a function of his utility function, and his 3) decision rule. Future work can focus on relaxing any of these three assumptions. The observer must infer anything that he does not initially know, while inferring θ simultaneously. The largest benefit-to-cost ratio probably comes from relaxing 2), followed by 1), followed by 3).

8 Appendix

8.1 The Fruit Machines

This section presents a thought experiment in a simple environment that is reminiscent of an n-armed bandit problem.

There are two vending machines that contain fruit. To operate a vending machine, the actor simply pulls its lever, which causes it to dispense an apple, a banana, or a cherry at random. Each vending machine has a fixed probability distribution of fruit that is known to the observer but not to the actor. The observer's goal is to infer the actor's preferences over fruit, represented as a utility function (see Definition 5) constrained to the interval $[0, 1]$. Suppose for concreteness that for fruits [apple, banana, cherry], the actor's utilities are $[1.0, 0.6, 0.0]$, the distribution of the first vending machine is $[0.5, 0, 0.5]$, and the distribution for the second vending machine is $[0, 1.0, 0]$. The observer knows the distributions of the vending machines but not the utilities of the actor. If the actor is an expert and chooses the vending machine that maximizes his expected utility, then he will always choose the second vending machine, from which the observer can only infer that the actor's utility for bananas is greater than the average of his utility for apples and cherries. On the other hand, suppose that the actor knows nothing about the fruit distributions of the vending machines a priori, updates his beliefs about the fruit distributions with Bayes' Rule, starting with a uniform prior, and always chooses the vending machine that maximizes his expected utility. Suppose that the following interaction history (see Definition 1) accumulates:

1. Choose Lever 1, obtain Apple
2. Choose Lever 1, obtain Apple

[13] This is not circular; the algorithm used to investigate other learning rules need not be efficient nor converge in general to collect some data.

3. Choose Lever 1, obtain Cherry
4. Choose Lever 1, obtain Cherry
5. Choose Lever 2, obtain Banana in every subsequent decision.

Now, what can the observer infer? It is helpful to make explicit the beliefs under which the actor made each decision, which are below. The ith line contains the actor's guess of the fruit distribution before making his ith decision.

1. $[1/3, 1/3, 1/3]$
2. $[2/4, 1/4, 1/4]$
3. $[3/5, 1/5, 1/5]$
4. $[3/6, 1/6, 2/6]$
5. $[3/7, 1/7, 3/7]$

First, since the actor chose to pull Lever 1 as their second action, he must have been happy with the Apple that he obtained the first time he pulled Lever 1. Thus, his utility for Apple is no less than the average of his three utilities. The second interesting decision is the fourth: the actor chose Lever 1 again, even after obtaining a Cherry. Although it may not be obvious why, this implies that the actor weakly prefers apples to bananas. To see why, let's see the constraint that this decision imposes on utilities U_A, U_B, and U_C:

$$\frac{3}{6}U_A + \frac{1}{6}U_B + \frac{2}{6}U_C \geq \frac{1}{3}(U_A + U_B + U_C) \tag{47}$$

where the left side is the expected utility obtained from pulling Lever 1, and the right side is the expected utility obtained from pulling Lever 2. This simplifies to

$$\begin{aligned} & 3U_A + U_B + 2U_C \geq 2U_A + 2U_B + 2U_C \\ \Rightarrow\ & U_A \geq U_B \end{aligned} \tag{48}$$

Finally, the fifth decision shows that the actor's utility for bananas is greater than the average of his utilities for apples and cherries, which can be shown through similar steps:

$$\begin{aligned} & \frac{3}{7}U_A + \frac{1}{7}U_B + \frac{3}{7}U_C \leq \frac{1}{3}(U_A + U_B + U_C) \\ \Rightarrow\ & 9U_A + 3U_B + 9U_C \leq 7U_A + 7U_B + 7U_C \\ \Rightarrow\ & 2U_A + 2U_C \leq 4U_B \\ \Rightarrow\ & U_B \geq \frac{1}{2}(U_A + U_C) \end{aligned} \tag{49}$$

From Equations 48 and 49, the observer can determine the actor's preference ordering over fruit:

$$U_A \geq U_B \geq U_C \tag{50}$$

This is much more information than the observer could ever hope to obtain from an expert actor.

9 References

[Abbeel and Ng, 2004] Abbeel, P. and Ng, A.Y., 2004, July. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning (p. 1). ACM.

[Austerweil *et al.*, 2015] Austerweil, J.L., Brawner, S., Greenwald, A., Hilliard, E., Ho, M., Littman, M.L., MacGlashan, J. and Trimbach, C., 2015. The Impact of Other-Regarding Preferences in a Collection of Non-Zero-Sum Grid Games.

[Benabou and Tirole, 2003] Benabou, R. and Tirole, J., 2003. Intrinsic and extrinsic motivation. The review of economic studies, 70(3), pp.489-520.

[Benhabib *et al.*, 2010] Benhabib, J., Bisin, A. and Schotter, A., 2010. Present-bias, quasi-hyperbolic discounting, and fixed costs. Games and Economic Behavior, 69(2), pp.205-223.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L., 2004. Convex optimization. Cambridge university press.

[Boylan and El-Gamal, 1993] Boylan, R.T. and El-Gamal, M.A., 1993. Fictitious play: A statistical study of multiple economic experiments. Games and Economic Behavior, 5(2), pp.205-222.

[Camerer and Ho, 1999] Camerer, C. and Hua Ho, T., 1999. Experience-weighted attraction learning in normal form games. Econometrica, 67(4), pp.827-874.

[Feldman and Serrano, 2006] Feldman, A.M. and Serrano, R., 2006. Welfare economics and social choice theory. Springer Science & Business Media.

[Gordon and Tibshirani, 2012] Gordon, G. and Tibshirani, R., 2012. Gradient descent revisited. Optimization, 10(725/36), p.725.

[Herman *et al.*, 2016] Herman, M., Gindele, T., Wagner, J., Schmitt, F. and Burgard, W., 2016, May. Inverse reinforcement learning with simultaneous estimation of rewards and dynamics. In Artificial Intelligence and Statistics (pp. 102-110).

[Howard, 1960] Howard, R., 1960. *Dynamic Programming and Markov Decision Processes*. The MIT Press. Cambridge, MA.

[Ng and Russell, 2000] Ng, A.Y. and Russell, S.J., 2000, June. Algorithms for inverse reinforcement learning. In Icml (pp. 663-670).

[Sutton and Barto, 1998] Sutton, R.S. and Barto, A.G., 1998. Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.

10 Acknowledgements

First and foremost, I want to thank Professor Amy Greenwald, whose ideas, generous time commitment, constant feedback, patience, and encouragement have been essential to this project.

I also want to thank Professor Michael Littman for being my official reader and also Professors George Konidaris and Shriram Krishnamurthi for coming to my defense. More generally, I want to thank all four of these professors for being such influential role models and advisors to me throughout my time at Brown.

Professor Paul Valiant and my friend Clayton helped me talk through a few of the proofs that I presented here. I thank them for their help.

My sister and my dad: thanks for coming to my defense. It meant a lot that you made the trip out here just to support me. Thanks also to my friends who could make it and endured listening to me talk constantly about this project for the past year.

Last but not least, I want to express appreciation for Nathan, who has helped to support me through and add carefree moments to the most challenging year in my career and one of the hardest years of my life so far.

I'm so grateful to have so many wonderful people in my life. Thanks, everyone!