Human-Actor Human-Critic:
Human Demonstrations and Human Feedback in the Action-Space

Jacob Beck
Brown University, Providence, RI
Jacob_Beck@Alumni.Brown.edu

In collaboration with
Naveen Srinivasan, Aansh Shah, Josh Roy

May 2020

Abstract

In many learning contexts, allowing an agent to explore new behaviors is prohibitively costly or dangerous. At the same time, supervised learning on data collected by experts only contains a narrow distribution of states and is subject to covariate shift at runtime. To mitigate this issue, we present an alternative framework, *Human-Actor Human-Critic (HAHC)*, wherein a human agent explores the environment and another human gives feedback. Using our framework, we are able to move our assumptions from the policy collecting the data, to the feedback on the data. Specifically, the demonstrations consists of actions and states, while the feedback consists of directions or distances in the same action space. We introduce the Directed HAHC framework (using directions) and the Undirected HAHC framework (using distances), analogous to Supervised Learning (SL) and Reinforcement Learning (RL), respectively. We propose and evaluate novel methods for learning in these two frameworks on synthetic data - showing the superiority of "margin-based approaches" - and begin an inquiry into the application of our methods on autonomous vehicles (AV).

1 Introduction

Requiring an agent to make mistakes, or even allowing mistakes, can be quite dangerous and costly in the real world. For this reason we investigate methods of training an agent without exploration, instead having a human actor collect the data. Supervised learning, which can allow a human actor to collect the data, has been explored for control, but it encounters the issue of the covariate shift. That is, training data collected from an optimal demonstration consists only of the states induced by the optimal control policy, but at runtime, the trained agent may encounter a vastly different state distribution and has seen little relevant training data. To mitigate this issue, we intentionally have our human actor deliberately explore the state-action space. However, as the actor reaches states off of the optimal trajectory, it must make sub-optimal decisions. In order to have our agent not replicate these suboptimal decisions, supervised learning requires that we either erase these actions, or replace these action with the correct action. Erasing these actions is wasteful and having another human, the critic, replace these actions is difficult, without the critic controlling the actor herself [Ross et al., 2012].

Since supervised learning falls short, we introduce two alternate frameworks, HAHC directed and HAHC undirected. Both frameworks use a dataset of states and actions, but unlike SL and RL, our frameworks make no assumptions about the joint distribution of states and actions. For example, SL assumes that actions are sampled from the optimal policy (generally with Gaussian noise, for Mean-Squared Error), whereas our HAHC frameworks allow for any distribution over states and actions.

HAHC moves assumptions on the distribution of data to the human critic. HAHC assumes that the feedback vector from the critic plus the action vector from the dataset, which together form a directed sample, is approximately correct, in that it is sampled about the optimal action. However, to deal with the difficulty of correcting actions the critic observes, we additionally assume that the bias and variance of our feedback vector is proportional to the distance between the actor's action in our dataset from the optimal action. In the HAHC directed case, we have the same assumptions, but the feedback in the dataset is the magnitude of the correction vector.

We investigate novel algorithms to operate in both HAHC frameworks. In the HAHC directed framework, each algorithm strictly generalizes SL. In the HAHC undirected framework, we draw analogies to RL. Evaluating on synthetic data, in both frameworks, we demonstrate that the best method is to view each piece of data as placing a constraint, in the form of a soft-margin, on the policy network. The soft margin ensures that we can learn from feedback with high bias and variance, without having them impact our learned policy.

Finally, we being an inquiry into the application of our methods to AV control. We find this application particularly compelling given that many humans are already trained to provide feedback in the action space. Moreover, a human critic can easily sit in the passenger seat with a secondary steering wheel to give feedback in real time. The data we collect is for the task of driving an AV while staying on the road (lane-following), with actions corresponding to steering angles, and this could easily be extended to incorporate other action such as acceleration and breaking. We term this application area "Backseat Driver". We present our initial inquiries and difficulties learning on this data, and present directions for future work on "Backseat Driving".

2 Related Work

Learning from demonstration (LfD) has mostly been studied in the SL framework and the Markov Decision Process (MDP) framework, as the basis for RL. In the former, it is generally known as imitation learning (or behavioral cloning in the AV setting). In the latter, it may arguable fall into inverse reinforcement learning (IRL) [Abbeel and Ng, 2004] or batch-constrained reinforcement learning (RL without exploration) [Fujimoto et al., 2018].

SL generally learns a regression that maps from an input state to an output action. For imitation learning, an expert performs an optimal demonstration, and then this data is used to train a policy network. A known issue with this approach is that the runtime and training distributions can be vastly different. That is, once the trained agent is acting on its own after training and encounters states it has not seen, it does not know how to act, and strays further from the intended behavior. This problem is known as the covariate shift and solutions generally follow the approach laid out in DAgger Ross et al. [2010]. DAgger allows the agent to explore and then uses the expert to label the new dataset, then training on all of the combined data. Such an approach has even been improved upon both to address scalar feeddback by incorporating experts that can label Q values with the AggreVaTeD algorithm [Ross and Bagnell, 2014], and to and to address deep neural networks by finding a policy gradient with the Deeply AggreVaTeD algorithm [Sun et al., 2017]. However, these DAgger based policies require the agent to explore and make mistakes in order to make incremental progress, which our HAHC frameworks intentionally avoid. Moreover, our HAHC frameworks makes use of non-optimal demonstrations as well.

The existing MDP research is not a great fit either. IRL generally tries to learn a reward function from expert demonstration and simultaneously optimize it [Abbeel and Ng, 2004, Shiarlis et al., 2016, Burchfiel et al., 2016]. Our work could arguably fall into the IRL category because we do NOT have a reward function that we can query on new data, and would have to learn it. However, we find that IRL is not a good fit for our problem setting: Should we learn a latent reward function, we cannot collect new data with additional experimentation, which may be dangerous and costly. And, more importantly, our HAHC framework does not assume we occupy an MDP with a termination condition, and the policy can be vastly different in repetitions of the same state. That is: we do not have an exploratory policy that is fixed over

time, nor do we have a proper MDP, with a termination condition, both of which present major issues for IRL.

Next, it is worth noting work on off-policy RL, starting with policy gradient methods. In off-policy policy-gradient methods data that the agent trains on does not come from the current learning policy. Unlike our problem setting, these algorithms all generally require that the exploratory policy have a non-zero probability of choosing any action in any state, that the exploratory policy is known, and that the exploration can be repeated at will (e.g. Off-Policy Actor-Critic [Degris et al., 2012], Q-Learning [Watkins and Dayan, 1992], Retrace [Munos et al., 2016], NAC [Gao et al., 2018]). Again, these methods require a proper MDP with a termination condition, which we do not have, and generally place requirements on exploration that make exploration dangerous, which we seek to avoid. Moreover, with the requisite human exploration, it is difficult to quantify the exact probability with which a human selected their action.

One subset of off-policy research – batch RL – is the best fit we have found. Fujimoto et al. [2018] propose the first deep RL algorithm capable of learning a continuous action from off-policy and fixed batch data. Their algorithm learns a probabilistic generative model and then uses a learned Q^* function to pick among them, in addition to other modifications to Q-learning intended to keep the state distribution of the learned policy similar to that of the expert. This approach is not viable as is, since we have no specific reward function, and even if we could craft one from our feedback without losing information, we have no notion of termination and episodes. Moreover, we do not want our learned policy to visit states similar to our sub-optimal demonstration. However, upon consideration, there is one reasonable method to apply to HAHC. Although we do not want to adjust our learned policy to reflect our given demonstrations, we note that the fact that our demonstrations are collected to sufficiently cover the state-action space, means that we should not need the additional changes introduced to Q-learning. That is, if we can use Q-Learning [Watkins and Dayan, 1992], and may not need to focus on the issue of covariate shift, as do Fujimoto et al. [2018]. The question still remains of how to get a reward function without a notion of termination and episodes. Instead of doing this, we find it compelling to compute a Q^* function directly in terms of our feedback, which already includes information about the optimal action given the long-term future, and then try to maximize Q^*. In doing so, we wind up learning to regress Q^* from samples of Q^* directly, without any bootstrapping (since we have no reward function). This is not an existing method, but is the closest fit to the RL literature, and so we will discuss this connection further and evaluate it in the undirected HAHC framework.

Out of all the LfD work specifically in the AV context, the most notable has either been on behavioral cloning [Bojarski et al., 2016, Pan et al., 2017] or using IRL to solve sub-tasks such as driving preferences that act on top of a safely functioning trajectory planner [Kuderer et al., 2015]. To the best of our knowledge, no research so far has focused on using any kind of evaluative feedback provided by a human in the context of AV control with LfD. That is, no one has solved how to take states, actions, and continuous feedback with respect to those actions, and convert them into a control policy for an AV, without having the AV explore. We believe that this is a major oversight: many AV research groups are investing huge amounts of time into collecting driving data; if they used our framework, they could improve performance simply by having a critic sit in the car with the driver for no additional real time. Thus we find motivation for "Backseat Driver" as an application of HAHC.

3 Framework

We consider a learning agent observing human demonstrations in a fully-observable environment, along with human feedback on those actions, in order to learn an optimal policy. We specify the data collection process as a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{P})$. At time-step t, the human actor inhabits some state, $s_t \in \mathcal{S}$, recorded into a dataset $\mathcal{D}$. The human takes some action $a_t \in \mathcal{A}$, also recorded in $\mathcal{D}$. The environment then transitions to state s_{t+1}. The agent learns a continuous *policy*, learning to predict an action conditioned on state $\pi(s_t)\colon \mathcal{S} \rightarrow \mathcal{A}$. This policy is learned to mean integrated squared error (MISE) to a deterministic optimal policy, $\mathcal{P}(s_t)\colon \mathcal{S} \rightarrow \mathcal{A}$. However, we do not have direct access to $\mathcal{P}$. Instead, we introduce a *feedback function* $\mathcal{F}$, modeling our human critic. Let $d_t = a_t - \mathcal{P}(s_t)$. At each time step, the human considers an intended

optimal action $\tilde{a}_t \sim \mathcal{P}(s_t) + \mathcal{N}(c_b\|d_t\| * u_t, c_v\|d_t\| * \mathcal{I})$, where $\mathcal{N}$ is a normal distribution, u_t is a unit vector, and $\mathcal{I}$ is the identity matrix. Note that $\tilde{a}_t$ has bias an variance proportional to $\|d_t\| = \|a_t - \mathcal{P}(s_t)\|$, with proportionality constants c_b and c_v, when used to estimate $\mathcal{P}(s_t)$. The human then converts the $\tilde{a}_t$ into feedback, $f_t = \tilde{a}_t - a_t$, which is stored in $\mathcal{D}$. Note that s_t, a_t, f_t are recorded in $\mathcal{D}$ together as (s, a, f), but the time and order in which they occur is not stored in $\mathcal{D}$. Also note that $f_t + a_t = \tilde{a}_t$, which would match the targets under standard SL assumptions, if instead $\tilde{a}_t \sim \mathcal{P}(s_t) + \mathcal{N}(0, c)$. Thus, instead of assuming the data are i.i.d., HAHC directed can be viewed as SL along with a measure of quality for each each target $\tilde{a}_t$, affecting bias and variance of that sample.

Although $\mathcal{D}$ is the same in the **directed setting** and **undirected setting**, we we place a further restriction on the undirected methods. Our methods in the **undirected setting** only can make use of $\|f_t\|$. Thus in the undirected setting, we cannot recover $\tilde{a}_t$, as we can in the directed setting by calculating $\tilde{a}_t = f_t + a_t$. We could equivalently redefine $\mathcal{D}$ and f_t for these two settings, but find it more convenient to place the restrictions on the methods than on the dataset so that we can store just one dataset.

As mentioned in section 2, our setting is off-policy, and this fact is difficult to properly account for within an RL framework, since we do not have a termination condition yielding a notion of episodes and returns. Specifically, converting feedback to rewards and calculating returns without a consistent termination condition is problematic: Although we do collect data along several different trajectories, the end of these trajectories is arbitrary and so an action that leads to large total reward in one episode may lead to small total reward in another. In order to side-step the issue of converting feedback to rewards and calculating returns without a consistent termination condition, we find it compelling to consider $\|f_t\|$ as a function of Q^*, since $\|f_t\|$ already includes information about optimal action given the long-term future. More specifically, we can view Q^* as a monotonically decreasing function of the magnitude of our feedback, in the **undirected setting**. That is, consider Q^* to be a function that is greater the smaller the necessary correction. It is important to note that given our fixed dataset, we have a fixed number of given queries to this Q^* function. This draws the closest RL analogy we find tenable.

4 Methods

All of our models share the same neural network architecture and optimizer (defined in the appendix). The only difference between them is the algorithm for determining the loss function. We have five main algorithms, four of which have a variant in both directed HAHC and undirected HAHC, and one of which exists only in the undirected framework. We describe these in detail in this section.

BC and BC Filtered. The first and second algorithm we consider are behavioral cloning (BC) and a generalization to the standard BC algorithm, which we term "BC Filtered". BC does not generally incorporate feedback and just performs a least-squares on the actions. Since the actions (a_t) we have in our dataset are not sampled from an expert distribution exclusively (and in fact have no structural restrictions), ignoring the corrective feedback and just performing a regression will not regress a function with any structure. The simplest fix to this algorithm is simply to filter out all actions with more than a certain correction. That is, if the magnitude of the corrective feedback is large, we ignore that action and regress the rest. By setting this filter parameter to infinity we can always recover standard BC. Note that if we are in the directer setting, we regress $\tilde{a}_t$ instead. Here, the purpose of the filter is solely to eliminate samples which, by assumption, have greater variance. Pseudo code is below:

```
def BC_Filtered_Loss(s, a, f, eps=inf):
    if |f| > eps:
        return 0 # Do NOT incorporate this sample into mean loss
    target = a+f if DIRECTED else a
    return (target - pi(s))^2
```

Variance. Our next algorithm makes use of the assumption that variance is proportional to the magnitude of the correction. In the normal derivation for MSE, it is assumed that the target values are sampled from the correct optimal action with Gaussian noise. However, the variance term is often left out and assumed to be absorbed into the learning rate, since this is what happens if the variance is constant for all data points. If, however, we know this variance, we can make use of it, deriving the following loss. Let θ be the model parameters. Our Maximum Likelihood Estimate (MLE) objective, for the one dimensional case, is:

$$Pr(data|model_\theta)$$

$$\prod_{s,a,d\in\mathcal{D}} \Pr(\tilde{a}|\mathcal{P}(s) = \pi_\theta(s), d)$$

Now, since we do not have d_t, we use $\|f_t\|$ as a reasonable estimator of $\|d_t\|$ (unbiased if $c_b = 0$):

$$\sum_{s,a,f\in\mathcal{D}} \log[\Pr(\tilde{a}|\mathcal{P}(s) = \pi_\theta(s), f)]$$

Thus,

$$Loss = -\log[\Pr(\tilde{a}|\mathcal{P}(s) = \pi_\theta(s), f)]$$

$$Loss = -\log(\frac{e^{-\frac{(\tilde{a}-\pi_\theta(s)+bias)^2}{2\sigma_f^2}}}{\sqrt{2\pi\sigma_f^2}})$$

$$Loss = \frac{(\tilde{a} - \pi_\theta(s) + bias)^2}{2\sigma_f^2} + \log(\sqrt{2\pi\sigma_f^2})$$

We remove the bias term ($c_b\|d_t\| * u_t \approx c_b\|f_t\| * u_t$), since, it can be absorbed into the $pi_\theta(s)$ term, as it may differ per input s. Thus,

$$Loss = \frac{(\tilde{a} - \pi_\theta(s))^2}{2\sigma_f^2} + \log(\sqrt{2\pi\sigma_f^2})$$

Generally, $\log(\sqrt{2\pi\sigma_f^2})$ can be left out since it is a constant w.r.t. our parameters and so will go away when the gradient is taken, leaving:

$$Loss = \frac{(y - \hat{y})^2}{2\sigma_f^2}$$

We can also leave out, $\frac{1}{2}$, since it only acts to scale the gradient, and can be accounted for by adjusting the learning rate of gradient descent, but crucially we do not leave out $\frac{1}{\sigma_f^2}$ leaving:

$$Loss = \frac{(y - \hat{y}_p)^2}{\sigma_f^2}$$

Now, by assumption $\sigma_f^2 \propto \|d_t| \approx \|f_t\|$:

$$Loss = \frac{1}{\|f\|}(\tilde{a} - \pi(s))^2$$

Thus we have a new algorithm that also generalizes SL, with no additional parameters. Specifically, if the corrections supplied are a constant for each datum, we recover BC. Note, for the undirected case, we replace $\tilde{a}$ with a. (This is no longer a rigorous derivation, but maintains similar interpretations and benefits.)

Interestingly, on-policy policy gradients can also be written Sutton et al. [1999]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\pi, a\sim\pi}[Q^\pi(s, a)\nabla_\theta \log(\pi_\theta(a|s))].$$

We cannot compute this gradient since we are never collecting data over a distribution induced by the learning policy π, but the form is similar enough to warrant a remark. Specifically, if we assume a Gaussian

π, and replace Q^π with Q^*, which we can interpret as the monotonically decreasing function of $\|f\|$, $\frac{1}{\|f\|}$, the we recover the same loss function inside the expectation. Thus, one way to get a handle on what the Variance model is doing, is to interpret it as using an on-policy gradient used off-policy. (We speculate that the positivity of $\frac{1}{\|f\|}$ is important here, since negative scaling of MSE will cause MSE to grow after each parameter update, thus diverging.) And yet another way to view this loss function is a re-weighting of the dataset to up-wight samples with corrective feedback that is small in magnitude. In any case, pseudo code for the algorithm follows:

```
def Variance_Loss(s, a, f):
    scale = 1/|f|
    target = a+f if DIRECTED else a
    return scale*(target - pi(s))^2
```

FNet. Our FNet method resembles the closest connection possible to the RL approach Q-Learning. Here, as mentioned, in order to side-step the issue of converting feedback to rewards and calculating returns without a consistent termination condition, we find it compelling to consider Q^* as a monotonically decreasing function of the magnitude of our feedback. That is, we learn to regress and maximise Q^* by learning to regress and minimize $\|f_t\|$. We not that there are no rewards (and thus no bootstrapping) and that this approach is inherently in the **undirected HAHC** framework. In order to predict $\|f_t\|$, we discretize the action range into 100 different bins, and append to the input the discrete action closet to the a_t taken. In order to select an action at runtime, we pick the discrete action from these bins that minimizes $\|f_t\|$, as in Q-Learning. Pseudo code for the loss is as follows:

```
def FNet_Loss(s, a_discrete, f):
    return (f - pi(s,a_discrete))^2
```

Margin. Our final method is a soft-margin approach encoding soft-constraints for each data-point. In **directed HAHC**, our directed margin defines a ball about $\tilde{a}$, for which there is no loss. The radius of this ball is determined the hyperparameter "critic_uncertainty" and (chosen by cross-validation) and is scaled by $\|f_t\|$. Beyond this sphere, actions are penalized by MSE to $\tilde{a}$. This loss thus recovers SL when all $\|f_t\| = 0$ and can be viewed as BC with an additional soft constraints defined by points with $\|f_t\| \neq 0$. The primary motivation here is that points with "lower quality", that is larger $\|f_t\|$, should not affect our learned policy at all, so long as were are relatively close to them, where this relative tolerance increases as the quality decreases. This stands in contrast to our Variance method, where points with large $\|f_t\|$ will always affect our loss to some extent, unless we predict them perfectly. Thus, we ensure some "slack" on data-points we are less sure about. For **undirected HAHC**, we again use $\|f_t\| \approx \|d_t\|$. Thus we know approximately how far $P(s_t)$ is from a_t. We thus can define a ring about a_t with zero loss. However, since $\|f_t\|$ is approximate, we make this ring into a shell with width proportional to our approximate quality, $\|f_t\|$. Outside of this ring, the error grows quadratically, as in MSE. Pseudo code for the final algorithm is below:

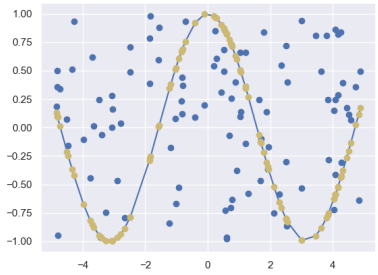
```
def Margin_Loss(s, a, f, critic_uncertainty):
    if DIRECTED:
        target = a+f
        tolerance = |f|*critic_uncertainty
        return max((target-pi(s))^2 - tolerance, 0)
    else:
        under_estimate = ((1-critic_uncertainty)*|f|)
        over_estimate = ((1+critic_uncertainty)*|f|)
        under_constraint = max(under_estimate^2 - (a-pi(s))^2,0)
        over_constraint = max((a-pi(s))^2 - over_estimate^2,0)
        return under_constraint + over_constraint
```

5 Synthetic Experiments

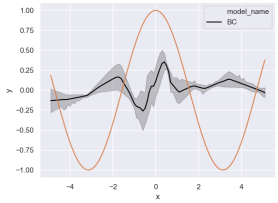
In order to evaluate our models, we regress a cosine function ($\mathcal{P} = cos(s)$) on the domain, $s_t \in [-5, 5]$. We sample s_t uniformly over this domain, and sample actions a_t uniformly over $[-1, 1]$. (These points are plotted in blue.) We then sample feedback, with $u_t = 1$, but c_b and c_v chosen differently for each of 3 settings, which we call "schemes". The point of these schemes is to asses how our methods respond to each of bias and variance in f_t. To visualize the data, $\tilde{a}_t$ is plotted in yellow given a state s_t, on the horizontal axis.

5.1 Scheme 1: Perfect Critic

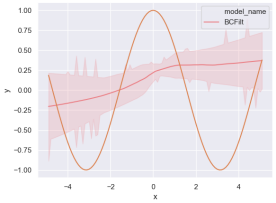
Although we ultimately want an algorithm to work with a biased and variant human critic, we want to ensure that our methods do not make training significantly worse under standard BC assumptions (which entails an expert demonstration). Thus we first set $c_b = 0$ and $c_v = 0$. A sample data set is visualized below:



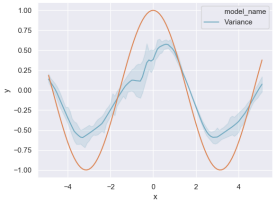
We visualize predictions for our models; rigorous tuning and evaluation is conducted in the next section:



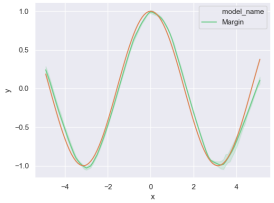
(a) Undirected BC



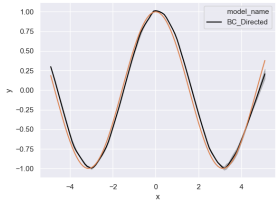
(b) Undirected BC Filtered



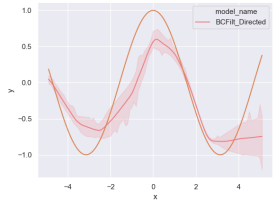
(c) Undirected Variance



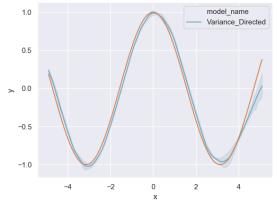
(d) Undirected Margin



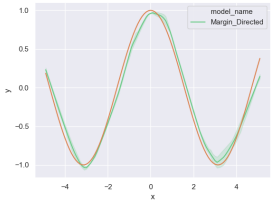
(e) Directed BC



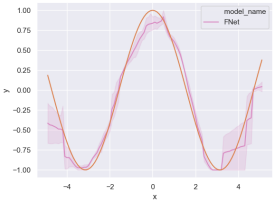
(f) Directed BC Filtered



(g) Directed Variance



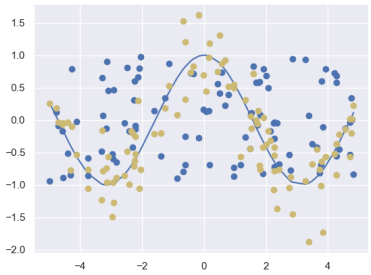
(h) Directed Margin

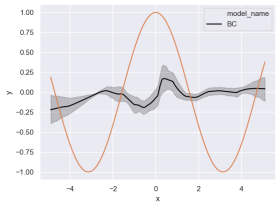


(i) FNet

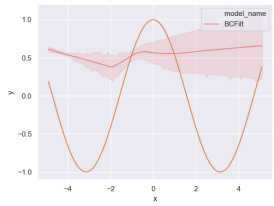
5.2 Scheme 2: Variant Critic

$c_b = 0$ and $c_v = 0.4$.

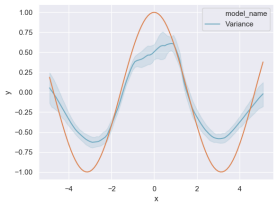




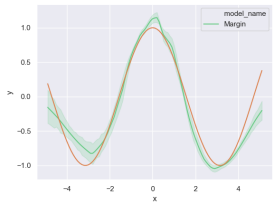
(a) Undirected BC



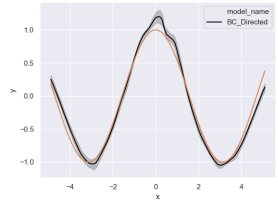
(b) Undirected BC Filtered



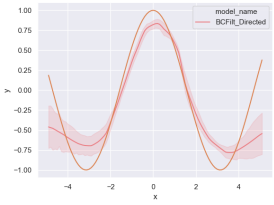
(c) Undirected Variance



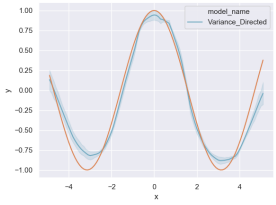
(d) Undirected Margin



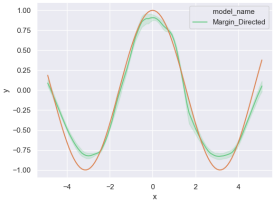
(e) Directed BC



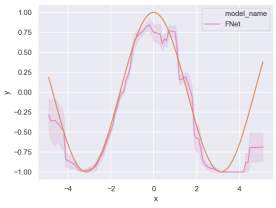
(f) Directed BC Filtered



(g) Directed Variance



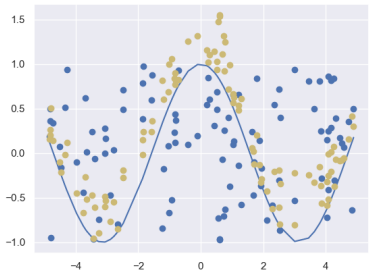
(h) Directed Margin

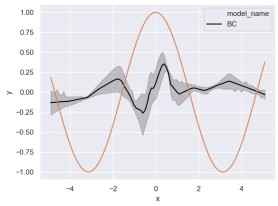


(i) FNet

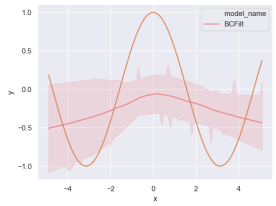
5.3 Scheme 3: Biased Critic

$c_b = 0.4$ and $c_v = 0$.

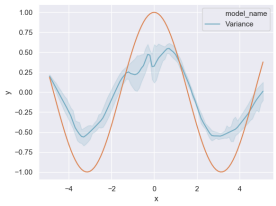




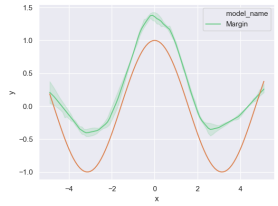
(a) Undirected BC



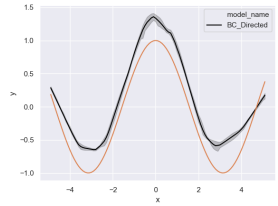
(b) Undirected BC Filtered



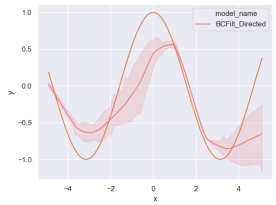
(c) Undirected Variance



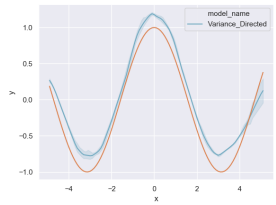
(d) Undirected Margin



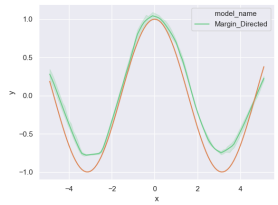
(e) Directed BC



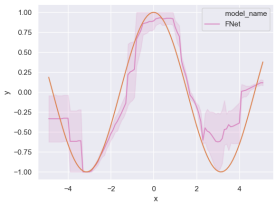
(f) Directed BC Filtered



(g) Directed Variance



(h) Directed Margin

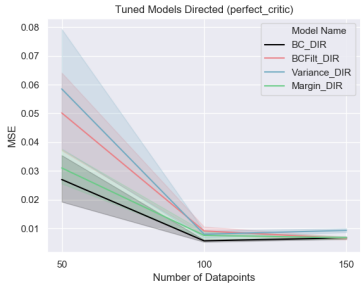


(i) FNet

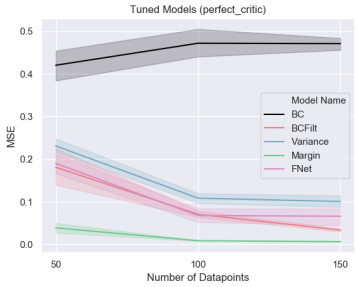
6 Synthetic Results

For each scheme, we train each model over three different amounts of data (50, 100, 150 data-points), and average MISE over schemes and amounts of data. We pick the best-hyperparameter setting, over a grid-search defined in the appendix. We then repeat this over a total of 5 seeds per model-parameter combination. We plot the results for each scheme and overall below in this section. (Lower is better.)

6.1 Scheme 1: Perfect Critic

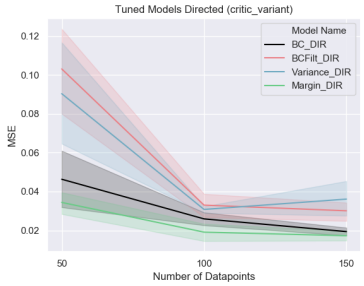


(a) Directed Models

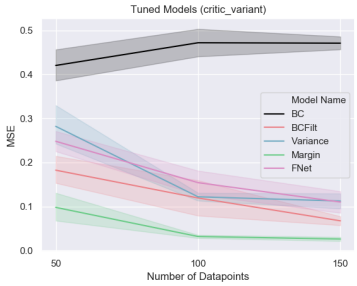


(b) Undirected Models

6.2 Scheme 2: Variant Critic

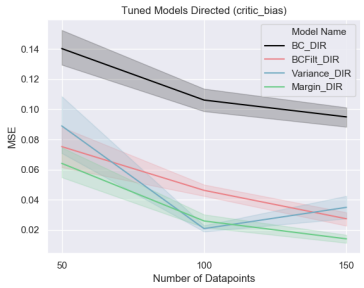


(a) Directed Models

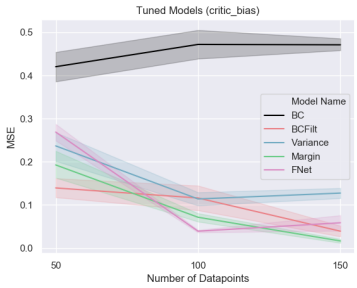


(b) Undirected Models

6.3 Scheme 3: Biased Critic

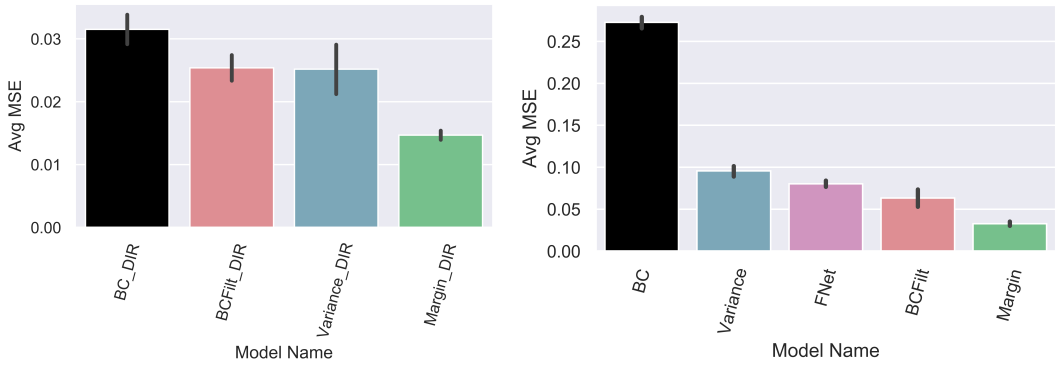


(a) Directed Models



(b) Undirected Models

6.4 Overall



(a) Directed Models

(b) Undirected Models

Our main findings on the synthetic data can be sumarized as follows

1. Margin methods, strictly dominate all other methods, indicating that they are successful in discounting effects of poor quality data.

2. BC in the undirected setting is a poor choice in method (as it ignores the feedback entirely) and learns a function unrelated to the intentions of the critic.

3. FNet performs on par with comparable undirected methods, but performs worse than all directed methods.

4. The Variance method performs much better than BC in the undirected setting, but does not perform well relative to other models. Moreover, it appears to perform the best relative to other methods in the biased scheme, despite being designed to take advantage of structure in the variance. In the variant scheme itself, we note that the Variance method has no particular advantage. That is especially surprising in the directed variant setting, where BC, which is the same method but does not approximate the variance, outperforms the Variance method itself. This indicates that the justification behind the Variance method does not hold up empirically. Approximating $\|d_t\|$ with $\|f_t\|$ may only be reasonable when the variance of f_t is reasonably low, which may defeat the purpose of estimating the variance in the first place. However, this same method may confer an advantage in that it down-weights biased samples.

5. BC Filtered performs surprisingly well for a method that just discards large swathes of data.

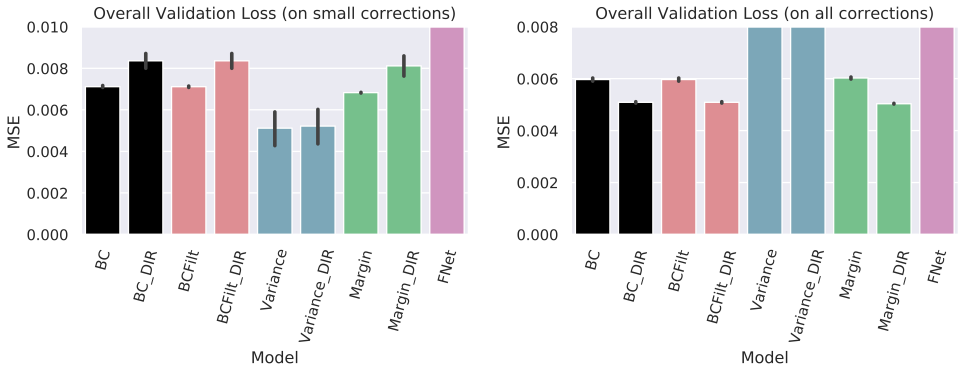
7 AV Inquiry

Here we introduce our preliminary inquiry into the application of HAHC to AV, which we term "Backseat Driver". Although our results are inconclusive, we present a summary of our work and difficulties.

In order to train our models using HAHC, we collected 60 minutes of driving data in simulation, having the human actor explore the state-action space for the task of lane-following. Intentionally, the human-actor took suboptimal actions, such as weaving and changing positions to the right and left of the road. We then had a human critic play back these actions in real time, and give feedback, corresponding to steering angle corrections, in the action-space. (More details on dara collection are included in earlier work Beck et al. [2019].)

The main difficulty in evaluation of our methods is that we do not have access to the true underlying optimal policy, $\mathcal{P}$. In order to to side-step this issue, we attempted to evaluate on data $\|f_t\| \approx 0$, since by assumption, such data is less biased. However, most of our data with small corrections falls within the expert state distribution, instead of outside the expert state distribution indicating how to "recover" from such states.

We still ran this flawed evaluation to see what would happen. Our results are inconclusive, suggesting that variance is significantly better when evaluating on data with small corrective feedback, but worse overall:



(a) Small Corrections

(b) All Corrections

One interesting results is that, using either metric, FNet performs far worse than all other methods, indication that it is not able to learn a reasonable model for the feedback. This could indicate that the feedback is hard to model without more data, or it could indicate that our feedback is too noisy.

8 Conclusion

We have proposed and motivated an alternate framework, Human-Actor Human-Critic (HAHC), for learning from human demonstrations and human feedback in the action-space. We have shown that our framework can be viewed as a generalization of SL, where we also have a measure of bias and variance, as determined by the magnitude of the feedback. Likewise, we have drawn analogies to RL, viewing feedback as a monotonically decreasing function in Q^*, with a limited number of queries. (As mentioned, FNet, and Variance to some extent, can be interpreted this way.) Moreover, we have proposed and evaluated novel methods on synthetic data, showing the superiority of margin-based approaches. Thus we see that our methods have much to gain by taking into account the structure of the action-space given through the feedback.

We hope that our undirected Margin method may be useful for many off-policy RL settings with Q^* and a limited number of queries. Answering whether this is true may come down to evaluating whether Q^* is generally related to a distance in the action space. Clearly there can be causes where this does not hold true: e.g. when an action with high Q^* is adjacent to an action with low Q^*. However, we note that in such as case, a standard policy-gradient with a Gaussian policy will not fair much better: the gradient will move the policy away from region of with low Q^π proportionally to how low Q^π is. Thus, we already have a distance assumption when using a Gaussian policy-gradient. Moreover, if the region with higher Q^* is included in our data, using our Margin approach, we can eliminate most of the effects of also having a data-point with lower Q^* in our data.

For future work, we will further investigate how these methods perform on noisy data collected in the real world to determine what happens when our modelling assumptions falter. Our AV data may be too noisy to use for experimentation, or it may be too difficult to evaluate. One potential way to more forward could be too evaluate in simulation rather than using our data, since this is where the problem of covariate shift will be most apparent. Additionally, we speculate that c_v may vary with s_t, thus we could infer $c_{v,t}$ from s_t, giving the margin method slack on data-points with small corrective feedback, and bringing its performance in line with that of the variance method on such data, while maintaining a reasonable performance on the dataset overall.

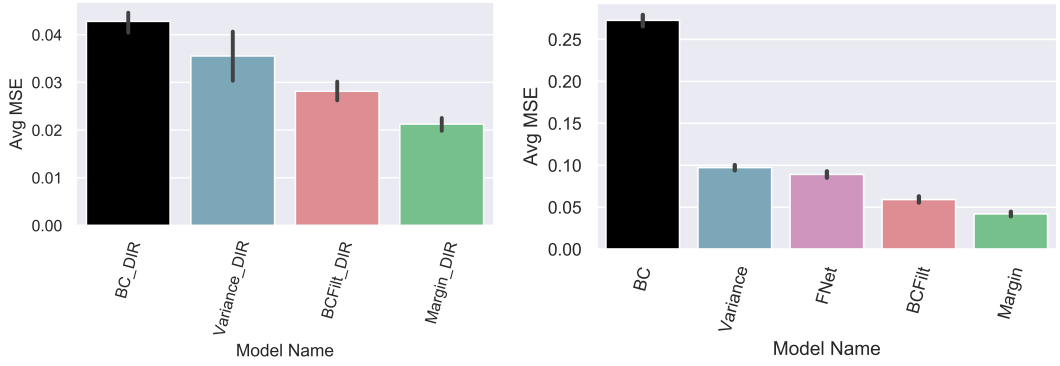
9 Acknowledgements

I would like to acknowledge Michael Littman, my advisor, for his continued advice and support over the years and for his for his wonderful sense of humor. I would also like to thank my collaborators, Naveen Srinivasan, Aansh Shah, Josh Roy.

10 Appendix

10.1 Proportionality Framework

In addition to the assumptions laid out in HAHC, we tested our synthetic data with the additional assumption that $f_t \propto \tilde{a}_t - a_t$, instead of $f_t = \tilde{a}_t - a_t$. In doing so, we modified each method to learn, via MLE, the to infer proportionality constant, c, with the estimate $\hat{c}$. All of the algorithms only change in that f is set to $\hat{c} * f$, before first use. Although not necessary for learning on our data, Results on synthetic data (with the true c set to 2 and the learned $\hat{c}$ initialized to 1 for gradient descent) are similar, and aggregated results are included for completeness:



(a) Directed Models

(b) Undirected Models

10.2 Hyper-Parameters

We use an Adam Optimizer. The network architecture for all experiments consists of three fully-connected layers of sizes: 10, 20, and 10. After these, we used a sigmoid to restrict the function's range. Note: This sigmoid is scaled by 2, so as to allow for actions near the boundary of our range (-1 and 1) without requiring sigmoid saturation. For hyper-parameter tuning, we use a grid-search.

Synthetic Data: For all models, we tune over 5 seeds, with 800 epochs and a batch size of 50, over the following learning rates: .1, .001, 0.00001.
Method Specific Parameters:
For the directed Margin method, we tune over critic uncertainties: 2.0, 0.8, 0.5, 0.2, 0.1.
For the undirected Margin method, we tune over critic uncertainties: 1.0, 0.4, 0.2, 0.1, 0.01.
For the directed and undirected BC Filtered method, we tune over filter epsilons: 1.0, 0.4, 0.2, 0.1, 0.0.

AV Data: For all models, we tune over 3 seeds, with 5 epochs and a batch size of 75, over the following learning rates: 1e-3, 1e-5, 1e-7. We also down-sampled 640x320 images to 40x30 using neighbor sampling (in addition to running experiments with convectional architecture from the Inception Network, at a resolution of 299x299). Our final model-specific hyper-parameters for tuning on the AV inquiry are in flux and will depend on our final evaluation metric.

References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *In Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

J. Beck, Z. Papakipos, and M. L. Littman. Reneg and backseat driver: Learning from demonstration with continuous human feedback. *CoRR*, abs/1901.05101, 2019. URL `http://arxiv.org/abs/1901.05101`.

M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL `http://arxiv.org/abs/1604.07316`.

B. Burchfiel, C. Tomasi, and R. Parr. Distance minimization for reward learning from scored trajectories. In *AAAI*, 2016.

T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. *CoRR*, abs/1205.4839, 2012. URL `http://arxiv.org/abs/1205.4839`.

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. *CoRR*, abs/1812.02900, 2018. URL `http://arxiv.org/abs/1812.02900`.

Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell. Reinforcement learning from imperfect demonstrations. *CoRR*, abs/1802.05313, 2018. URL `http://arxiv.org/abs/1802.05313`.

M. Kuderer, S. Gulati, and W. Burgard. Learning driving styles for autonomous vehicles from demonstration. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646, 2015.

R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. *CoRR*, abs/1606.02647, 2016. URL `http://arxiv.org/abs/1606.02647`.

Y. Pan, C. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots. Agile off-road autonomous driving using end-to-end deep imitation learning. *CoRR*, abs/1709.07174, 2017. URL `http://arxiv.org/abs/1709.07174`.

S. Ross and J. A. Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *CoRR*, abs/1406.5979, 2014. URL `http://arxiv.org/abs/1406.5979`.

S. Ross, G. J. Gordon, and J. A. Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010. URL `http://arxiv.org/abs/1011.0686`.

S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive UAV control in cluttered natural environments. *CoRR*, abs/1211.1690, 2012. URL `http://arxiv.org/abs/1211.1690`.

K. Shiarlis, J. Messias, and S. Whiteson. Inverse reinforcement learning from failure. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 1060–1068, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-4239-1. URL `http://dl.acm.org/citation.cfm?id=2936924.2937079`.

W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. *CoRR*, abs/1703.01030, 2017. URL `http://arxiv.org/abs/1703.01030`.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1999.

C. J. C. H. Watkins and P. Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992.