Sampling-based Maximum Entropy IRL

Kieron Kretschmar
University of Amsterdam
Student no. 11133139

Abstract

Maximum Entropy (MaxEnt) IRL attempts to learn a reward function from expert demonstrations while maximizing the entropy of decisions when the demonstrations are ambiguous. This method relies on gradient optimization. One component of the gradient is an expectation over the distribution of trajectories implied by the current reward model. The authors of MaxEnt IRL propose an approximation inspired by value-iteration algorithm that requires a small state-space but is potentially intractable otherwise. In this project, we propose sampling-based alternatives to obtain that approximation. We run experiments in an IcyGridWorld environment to evaluate inhowfar these methods can replace the original approach. Our results show qualitative differences in the learned reward functions. Finally, we propose directions for future research to investigate the strengths and weaknesses of these approaches in more detail.

1 Introduction

A common problem in Reinforcement Learning (RL) is to learn a policy π that chooses actions based on a state in a Markov Decision Process (MDP) such that a reward is maximized. Commonly, these rewards are modelled as part of the environment and, for each timestep, depend on the action chosen by the agent and the next state that the agent ends up in.

Choosing a reward model that accurately captures the designer's intentions can be quite difficult. For example, work on *Specification Gaming* [7] has shown that agents trained on a non-ideal reward model can find unforeseen solutions that gain high reward despite being undesired by its creators. Another failure mode is *Goal Misgeneralization* [6], where agents retain capabilities when switching from training to testing but pursue the wrong goal. To make matters worse, Goal Misgeneralization has been shown to occur even when the specifications were correct [11].

Methods from *Imitation Learning* (IL) circumvent this difficulty of specifying a reward function by leveraging *expert demonstrations*, usually generated by humans capable of solving the task. The simplest example of IL techniques is perhaps Behavior Cloning, which aims to learn a policy that mimics the demonstrated behavior. However, Behavior Cloning has severe limitations, particularly when the demonstrations only cover part of the state-space or time horizons are long.

Inverse Reinforcement learning (IRL) is another IL technique, which aims to *learn* the reward model from such demonstrations, which can subsequently be used to train a policy [10]. The assumption is that for some problems it is much easier for humans to demonstrate examples of good behavior than it is to explicitly specify a reward function, and that a good reward function can be extracted from these demonstrations. Notably, through their demonstrations, IRL puts humans into the loop of the learning process, whereas traditional RL does not. One perspective of why this may be advantageous is that the learning signal can be obtained from a distribution of multiple persons and, therefore, be hopefully more robust than if it was designed just by the person(s) developing the model.

Results from [1] show that when the reward function is linear in the state-features, then a policy optimizing for such a reward function is equivalent to the policy matching the expected distribution

of features visited as the experts. However, this constraint is usually not enough to specify a unique reward function. Ziebart et al. introduce *Maximum Entropy IRL* (MaxEnt IRL) to resolve this ambiguity by employing the principle of *maximum entropy* [12]. The motivation is to uniquely select the (stochastic) policy which matches the feature expectations while assigning equal probability to all trajectories that obtain equal rewards.

The main contribution of this project is to provide an adaptation of the original MaxEnt IRL algorithm. This change relaxes an assumption on the underlying MDP made by [12], discussed in section 3. Thereby, it makes MaxEnt IRL applicable to additional cases compared to the original method.

In section 4, we develop experiments to address the question to what extent sampling-based methods can recover the true reward as well or better than MaxEnt IRL in its original form. In section 5 we discuss the results.

2 Background & Related Work

In MaxEnt IRL the reward that a trajectory $\tau = \{(s_1, a_1), ...s_{T-1}, a_{T-1}, s_T\}$ of length T obtains is modelled linearly in the features as

$$R_\theta(\tau) = \theta^T \mu_\tau,$$

where θ is a vector containing the reward function's parameters and $\mu_\tau \coloneqq \sum_{s \in \tau} \mu(s)$ is the state-visitation frequency of τ.

Assuming a deterministic environment, the probability of a trajectory τ being generated under this model is proportional to the exponentiated reward obtained [12], so

$$p(\tau|\theta) = \frac{1}{Z(\theta)} \exp(R_\theta(\tau)) = \frac{1}{Z(\theta)} \exp(\theta^T \mu_\tau), \tag{1}$$

where $Z(\theta) = \int_\tau \exp(\theta^T \mu_\tau) d\tau$ is the normalizing constant in the form of an integral over all possible trajectories τ. Note that in a non-deterministic environment we have to take the environment's transition function $tr(s_{t+1}|a_t, s_t)$ into account, leading to a modified expression for $p(\tau|\theta, T)$.

Ziebart et al. show that fitting the reward function's parameters θ to N experts' trajectories $\{\tau_{E,n}\}_{n=1,...,N}$ can be done by maximizing the data-loglikelihood [12]

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}}\, \frac{1}{N} \sum_{n=1}^{N} logP(\tau_n|\theta, T). \tag{2}$$

The solution to this problem can be computed by gradient ascent optimization

$$\theta_{new} \leftarrow \theta_{old} + \alpha \nabla_\theta L(\theta) \tag{3}$$

with the learning rate α and the gradient

$$
\begin{aligned}
\nabla_\theta L(\theta) &= \nabla_\theta \frac{1}{N} \sum_{n=1}^{N} \log P(\tau_{E,n}|\theta, T) \\
&= \nabla_\theta \frac{1}{N} \sum_{n=1}^{N} \log(\frac{1}{Z(\theta)} \exp(R_\theta(\tau_{E,n}))) \\
&= \nabla_\theta \frac{1}{N} \sum_{n=1}^{N} (R_\theta(\tau_{E,n}) - \log(Z(\theta))) \\
&= \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta R_\theta(\tau_{E,n}) - \frac{1}{N} \nabla_\theta \log(Z(\theta)) \\
&= \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta \theta^T \mu_{\tau_{E,n}} - \frac{1}{Z(\theta)} \log(\nabla_\theta Z(\theta)) \\
&= \frac{1}{N} \sum_{n=1}^{N} \mu_{\tau_{E,n}} - \frac{1}{Z(\theta)} \int_\tau \nabla_\theta \exp(\theta^T \mu_\tau) d\tau \\
&= \frac{1}{N} \sum_{n=1}^{N} \mu_{\tau_{E,n}} - \int_\tau \frac{\exp(\theta^T \mu_\tau)}{Z(\theta)} \mu_\tau d\tau \\
&= \frac{1}{N} \sum_{n=1}^{N} \mu_{\tau_{E,n}} - \int_\tau p(\tau|\theta) \mu_\tau d\tau \\
&= \frac{1}{N} \sum_{n=1}^{N} \mu_{\tau_{E,n}} - \mathbb{E}_{\tau \sim p(\tau|\theta)}[\mu_\tau].
\end{aligned}
$$

The experts' average state-visitation frequency on the left is easily computed given their trajectories. Unfortunately, we generally can not sample from p directly, so we can not simply rely on a Monte-Carlo approximation for the expectation

$$
\mathbb{E}_{\tau \sim p(\tau,\theta)}[\mu_\tau]. \tag{4}
$$

In the remainder of this section we will investigate different approaches for approximating Equation 4.

In *Algorithm 1* of the original MaxEnt IRL paper, Ziebart et al. propose to approximate Equation 4 by running $N_Z \in \mathbb{N}$ iterations of an algorithm similar to forward-backward algorithm for Conditional Random Fields or value iteration [12] in RL. They obtain good results, but there are some limitations in applicability, because the forward pass of the algorithm approximates components of $Z(\theta)$. More precisely, a summation over all possible states and actions is performed, which implies assumptions on the MDP: If the state and action spaces are too large, or even continuous, this may not be possible. Further, the algorithm requires the transition dynamics to be known, which is a typical limitation of the value-iteration algorithm.

Guided Cost Learning proposed by Finn et al. [3] offers an alternative approach to simultanously approximate $Z(\theta)$ and optimize a policy in an iterative fashion. In each iteration trajectories are sampled from a physical system, which are used to both improve the estimate of $Z(\theta)$ and the policy. The idea is that the updated policy will yield trajectories in the next iteration that are of higher importance for approximating the partition function. This sample-based approach requires neither the transition dynamics to be known, nor a small state space.

Further methods to estimate $Z(\theta)$ have been proposed [4] [2] [8] but are beyond the scope of this project.

Our method serves as an alternative and will be described in detail in the next section.

3 Method(s)

In the following, we will assume that the partition function $Z(\theta)$ is computationally intractable, which makes the approach of [12] unfeasible. Instead, we will apply (self-normalized) *importance sampling* [5] in conjunction with a proposal distribution $q(\tau)$, from which we can sample, to approximate Equation 4 in the following way:

$$\begin{aligned}
\mathbb{E}_{\tau\sim p(\tau,\theta)}[\mu_\tau] &= \mathbb{E}_{\tau\sim q(\tau)}[\frac{p(\tau,\theta)}{q(\tau)}\mu_\tau] \\
&= \frac{\mathbb{E}_{\tau\sim q(\tau)}[\frac{p(\tau,\theta)}{q(\tau)}\mu_\tau]}{\mathbb{E}_{\tau\sim q(\tau)}[\frac{p(\tau,\theta)}{q(\tau)}]} \\
&= \frac{\mathbb{E}_{\tau\sim q(\tau)}[\frac{\tilde{p}(\tau,\theta)}{q(\tau)}\mu_\tau]}{\mathbb{E}_{\tau\sim q(\tau)}[\frac{\tilde{p}(\tau,\theta)}{q(\tau)}]} \\
&\approx \frac{\frac{1}{N}\sum_{n=1}^{N}\frac{\tilde{p}(\tau,\theta)}{q(\tau_n,\theta)}\mu_{\tau_n}}{\frac{1}{N}\sum_{n=1}^{N}\frac{\tilde{p}(\tau,\theta)}{q(\tau_n,\theta)}} \\
&= \sum_{n=1}^{N}\frac{w_n}{\sum_{j=1}^{N}w_j}\mu_{\tau_n}
\end{aligned}$$

with the importance weights

$$w_n \coloneqq \frac{\tilde{p}(\tau,\theta)}{q(\tau)}. \tag{5}$$

The probability of a trajectory $\tau = \{(s_1, a_1), ...s_{T-1}, a_{T-1}, s_T\}$ of length T, generated by a policy $\pi(a|s)$ and a transition function $tr(s_{t+1}|a_t, s_t)$, factorizes as

$$p(\tau) = p(s_1)\prod_{t=1}^{T}\pi(a_t|s_t)tr(s_{t+1}|s_t, a_t). \tag{6}$$

Note that the appearance of the transition function does not pose a problem for importance sampling for non-deterministic problems in which the transition function is unknown, which is an essential insight to perform *off-policy* RL. This is due to the transition probabilities cancelling out when computing the importance weights, which we can see when we denote the policy implicitly defined by p as π_p, and π_q analogously:

$$\begin{aligned}
w_n &\coloneqq \frac{\tilde{p}(\tau,\theta)}{q(\tau)} \\
&= \frac{p(s_1)\prod_{t=1}^{T}\pi_p(a_t|s_t)tr(s_{t+1}|s_t)}{p(s_1)\prod_{t=1}^{T}\pi_q(a_t|s_t)tr(s_{t+1}|s_t).} \\
&= \frac{\prod_{t=1}^{T}\pi_p(a_t|s_t)}{\prod_{t=1}^{T}\pi_q(a_t|s_t)}.
\end{aligned}$$

3.1 Proposal distributions

In principle, we are free in the choice of $q(\tau)$, as long as

1. we can sample trajectories from $q(\tau)$,
2. we can evaluate $q(\tau)$ up to a normalization constant,
3. $supp(p) \subseteq supp(q)$, as that is required to make the estimator of Equation 4 consistent.

That being said, we would expect the approximation to converge faster the closer our proposal q is to p.

In the remainder of this subsection we will introduce the different proposal distributions we will later evaluate.

One approach is to sample trajectories by rolling out a policy π_{random} that selects actions uniformly. However, using this distribution leads to a practical issue when trajectories have different lengths. To see this, we assume n_{action} possible actions per state and a deterministic environment for simplicity. Now, we observe that the probability of a trajectory will be $q(\tau) = (\frac{1}{n_{action}})^T$, which goes towards 0 exponentially fast in the length of the trajectory $|\tau|$. As that behaviour does not apply to the target distribution from Equation 1, the importance weights from Equation 5 will be dominated by those corresponding to the longest trajectories. Early experiments have shown that the gradients diverge within few iterations. Therefore, we have decided to leave this approach out of the experiments in the following sections.

One possible remedy for this effect we evaluate is to assign equal probabilities to all trajectories generated by π_{random}. We will denote the resulting distribution of trajectories by $q_{uniform}(\tau)$. Note that Finn et al. have resorted to a similar strategy [3].

Another approach that should provide a proposal distribution closer to the target $p(\tau|\theta)$ is to actually train a policy with the reward based on the current parameters θ, leading to a distribution we refer to as $q_{trained}(\tau)$. For this, we apply q-learning and extract the policy from the learned q-values by applying the softmax function. Here, we experience the same issue with trajectories of different lengths, so here, too, we will assign equal probabilities to all trajectories generated by $\pi_{trained}$.

3.2 Baselines

We will compare our results against two baselines, which are briefly described below. Note that neither of them correspond directly to any proposal distribution, as they are outside the importance sampling framework.

For the first baseline, $baseline_{mc}$, we will naively use the Monte Carlo estimate for Equation 4 with samples obtained by rolling out π_{random}. As this estimate is independent of the parameter θ, we do not expect to obtain good results.

For the second baseline, we will consider setting Equation 4 to a uniform distribution over all states, which we will simply denote by $baseline_{uniform}$. We do this not because we expect it to work well, but rather to have a baseline that should be beaten by any reasonable solution.

4 Experiments

We will perform experiments to evaluate our (importance-)sampling-based approach in an attempt to answer the research question introduced at the end of section 1.

To this end, we build upon an existing implementation [9] of MaxEnt IRL. Our implementation and experiments can be found on github [1].

4.1 The MDP: IcyGridWorld

The MDP we will evaluate our method on is called *IcyGridWorld*. It is a 5x5 GridWorld in which the agent starts the game in the bottom-left corner, and always has 4 actions to choose from, one for each direction left/right/up/down. When the agent navigates into a wall, its state will not change. The agent gets rewarded for navigating to 2 specific cells, highlighted in Figure 1. The cell with the highest reward is located in the top-right corner and simultaneously serves as the only terminal state. The *Icy* in the name comes from the additional property that agents in this world have a chance to slip with probability $p_{slip} = 0.2$ whenever they choose an action. When this happens, the agent's action will be ignored and instead it ends up in a randomly chosen neighboring state. The feature of each of the 25 states is the 25-dimensional 1-hot encoded vector, which entails that the reward parameters θ are a 25-dimensional real-valued vector.

[1] https://github.com/KieronKretschmar/sampling-based-maxent-irl

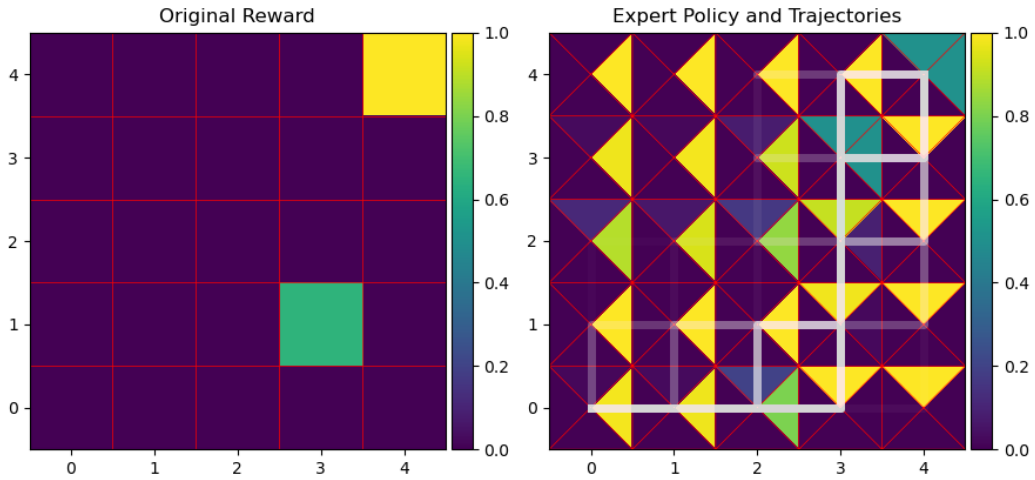

Figure 1: Left: IcyGridWorld and the original reward function we wish to recover. Right: Triangles in each state indicate the action chosen by the expert policy, which was created based on the true reward with value-iteration. 200 expert demonstrations are shown as transparent white lines.

4.2 Expert demonstrations

To apply any inverse RL technique we require expert trajectories from which we can learn a reward function. Because this setting is so simple, we can run the value-iteration algorithm and derive an optimal stochastic policy. Unrolling this policy provides us with the trajectories we require. Note that this resembles a setting where the experts act optimally, which is not always the case in IRL. Note that the trajectories are not guaranteed to be ideal due to the risk of slipping modeled in this world.

4.3 Hyperparameters

For our experiments we use the hyperparameters in Table 1, unless explicitly stated otherwise.

Hyperparameter	Default value
Learning rate in Equation 3 at iteration k	$\alpha = \frac{0.2}{1+k}$
MDP (IcyGridWorld)	
The probability of a random action being executed	p_{slip}=0.2
Discount factor γ_{VI}	0.9
Expert trajectory generation	
Probability of expert's action	$p(s_t, a_t) = \frac{v(s_{t+1}\|a_t)^w}{\sum_a v(s_{t+1}\|a)^{50}}$
Number of expert trajectories	200
Number of iterations for Algorithm 1 from Ziebart et al.	$N_Z = 2n_{states} = 50$
Training for $q_{learned}$	
Learning rate	$\alpha_q = 0.1$
Exploration probability	$p_{explore} = 0.1$
Training iterations	100

Table 1: Hyperparameters used in all our experiments.

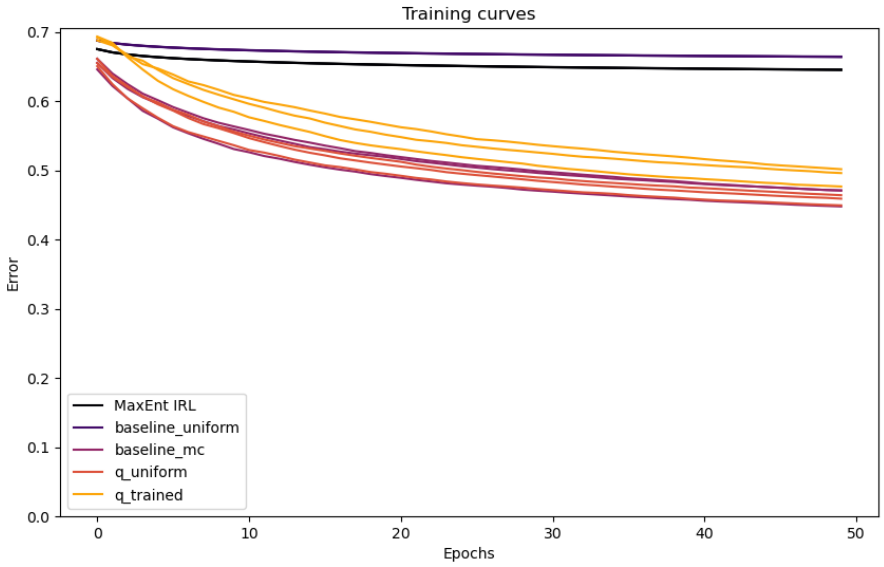

Figure 2: Training of the different approaches of MaxEnt IRL. Each epoch on the x-axis corresponds to one gradient update according to Equation 3. The error is computed as the L2-norm of the difference between the true and the learned reward parameter θ. Due to the scale-invariance of reward models in RL, we have normalized both vectors prior to comparison. We used 3 different seeds and expert trajectories for each approach. The highest errors are obtained by MaxEnt IRL and $baseline_{uniform}$, which are rather consistent across seeds. The curves of $q_{trained}$ lie a bit above those of $baseline_{mc}$ and $q_{trained}$, which are very close to each other and perform best according to this definition of error.

4.4 Results

In Figure 2 we show the difference between the true reward and the learned rewards from applying MaxEnt IRL in the original form as proposed by [12], our methods, and baselines as described above. Surprisingly, the learned parameters of the reward function θ are closest to the true values for $baseline_{mc}$ and $q_{uniform}$, despite the limitations on the baseline mentioned in subsection 3.2. We are equally surprised by the original MaxEnt IRL implementation obtaining such a high error, barely better than $baseline_{uniform}$. The errors for $q_trained$ are located between these extremes, suggesting that the procedure of training a policy based on the current estimate of θ in each iteration is not worth its sizeable computational cost. A brief discussion on the limitations of evaluating methods this way is provided in section 5.

To get a better understanding of what the different strategies actually learn, we have visualized the learned reward models in Figure 2. We observe that MaxEnt IRL assigns rather smooth rewards to locations close to where the true reward is located. In contrast, the sampling based methods all display very similar behaviour to each other, in that they all estimate almost all value in the top-right corner, which has the highest true reward and simultaneously serves as terminal state.

The estimate obtained from $baseline_{uniform}$ looks like it highlights the direct path(s) from start (bottom-left) to the terminal state (bottom-right). This matches our expectation, as in this case Equation 4 is uniformly distributed over all states and the gradients from Equation 3 are now pushing the reward towards the average state-visitation frequency of the experts' trajectories. This explanation is supported by observing that the expert trajectories visualized in Figure 1 closely match the reward learned by $baseline_{uniform}$.

Another dimension to investigate is the sample efficiency of these methods, i.e. how many expert demonstrations they require to obtain reasonable results. In Figure 4 we show the error in the learned reward each method obtains with varying numbers of demonstrations. Surprisingly, we observe that

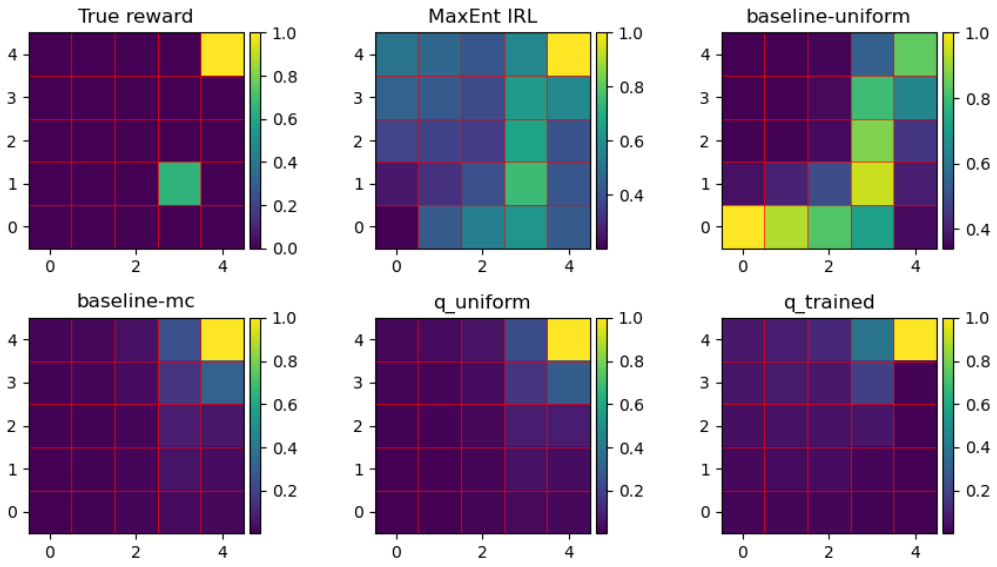

Figure 3: The normalized rewards learned after 50 epochs. We use the same setup as in Figure 2.

the performance of all methods is almost constant with regards to the number of expert trajectories when this number lies between 20 and 200. Looking back at the expert trajectories visualized in Figure 1, we suspect that this problem is simply too easy with expert trajectories showing almost no variance. This would explain why a few of such demonstrations are enough to sufficiently capture the experts' intentions.

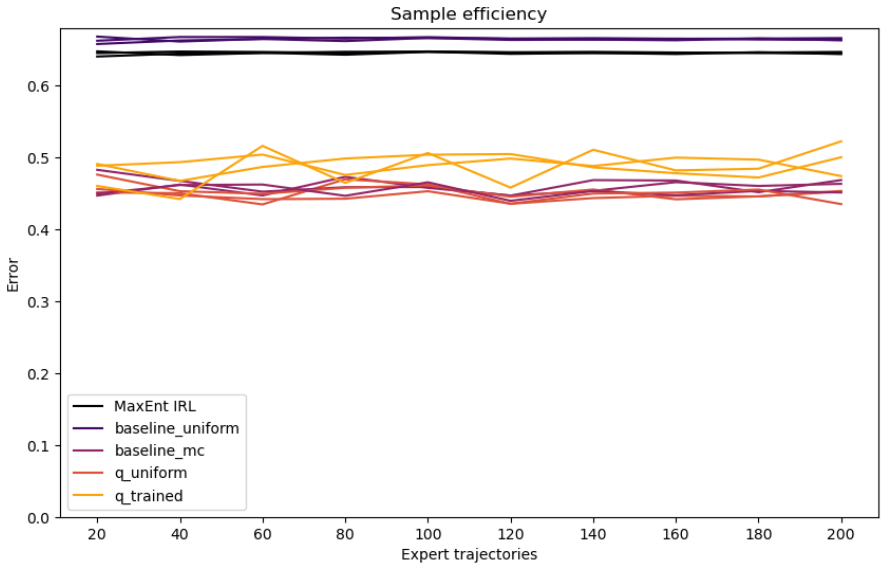

Figure 4: Methods evaluated with different numbers of demonstrations. Surprisingly, the number of demonstrations seems to have little to no impact on the error. Except for the number of demonstrations, the setup is identical to Figure 2.

5 Conclusions, Limitation, & Future Work

In this work we have investigated the applicability of different sampling-based methods in the MaxEnt IRL framework. These methods substitute the *value-iteration*-like algorithm proposed by [12]. The main advantage with using sampling-based methods for this is that they do not necessarily require a small state-space, which makes them potentially more applicable to a wider range of MDPs.

We evaluated the methods in multiple experiments carried out in a GridWorld environment. In our opinion, the results do not conclusively answer the question of whether sampling-based methods recover the reward as well or better than MaxEnt IRL in its original form. While sampling-based methods obtain a lower error as shown in Figure 2, we doubt whether this measurement adequately captures the performance of the algorithms. Rather we observe that the sampling-based methods find qualitatively different solutions: The original MaxEnt IRL estimates a rather smooth reward signal, whereas the sampling-based methods seems to place most emphasis on the state with the highest reward. One interpretation is that smooth reward signals are closer to the fundamental goal of maximizing entropy that MaxEnt IRL follows. We encourage future work aiming to evaluate the learned rewards in a more principled way.

With regards to the sampling-methods we use, we would encourage further exploration of ways to handle the problems related to dominant importance weights stemming from long trajectories as discussed in subsection 3.1.

More experiments in diverse environments are needed to thoroughly evaluate the strengths and weaknesses of the different approaches. This includes environments in which rewards are differently distributed and the state with the highest reward is not the only terminal state, to make sure that sampling based methods actually find the highest reward and not just locate the terminal state. Additionally, experiments in environments with large state-spaces are needed to validate whether sampling-based methods actually perform well where the original MaxEnt IRL approach fails. Lastly, more thorough evaluation with regards to sample efficiency is needed, as in our environment the number of demonstrations has almost no impact at all (as shown in Figure 4), which we attribute to the expert trajectories being very similar. We suspect that the last point may be addressed by simply adding noise to the experts' demonstrations, which could provide an easily implementable extension to this project.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, New York, NY, USA, 2004. Association for Computing Machinery.

[2] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 182–189, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[3] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 49–58, New York, New York, USA, 20–22 Jun 2016. PMLR.

[4] D.-A. Huang and K. M. Kitani. Action-reaction: Forecasting the dynamics of human interaction. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 489–504, Cham, 2014. Springer International Publishing.

[5] T. Kloek and H. K. van Dijk. Bayesian estimates of equation system parameters: An application of integration by monte carlo. *Econometrica*, 46(1):1, Jan. 1978.

[6] J. Koch, L. Langosco, J. Pfau, J. Le, and L. Sharkey. Objective robustness in deep reinforcement learning. *CoRR*, abs/2105.14111, 2021.

[7] V. Krakovna, J. Uesato, V. Mikulik, M. Rahtz, T. Everitt, R. Kumar, Z. Kenton, J. Leike, and S. Legg. Specification gaming: the flip side of ai ingenuity., 2020.

[8] S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples, 2012.

[9] M. Luz. Maximum Entropy Inverse Reinforcement Learning - An Implementation, https://github.com/qzed/irl-maxent, July 2019.

[10] A. Y. Ng, S. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[11] R. Shah, V. Varma, R. Kumar, M. Phuong, V. Krakovna, J. Uesato, and Z. Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals, 2022.

[12] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438, 2008.