Methods of Bayesian Inverse Reinforcement Learning

William Lin
EECS MS
UC Berkeley
willin@berkeley.edu

Alina Trinh
EECS MS
UC Berkeley
tutrinh@berkeley.edu

Abstract

We present a survey paper and comparisons of three Bayesian Inverse Reinforcement Learning (BIRL) methods: the original BIRL method introduced in [18], Bayesian Reward EXtrapolation (B-REX) introduced in [7], and Approximate Variational Reward Imitation Learning (AVRIL) introduced in [8]. The original method and B-REX generate a candidate reward using Markov Chain Monte Carlo (MCMC) methods on the posterior reward distribution, while the AVRIL method uses a neural network as a function approximator to give a candidate distribution for the rewards. We also include experimental results that show how each method performs regarding sample complexity and distance from the ground truth reward upon convergence.

1 Introduction

Inverse Reinforcement Learning (IRL) is a subset of reinforcement learning that deals with finding a reward function for states given demonstrations of a task.

When deploying AI systems or robots in practice, there are often many problems or tasks for which it is difficult for the human to specify an exact reward function for the AI to follow. Often times design of reward functions can be complicated, and improper design can lead to reinforcement learning algorithms exploiting errors and producing suboptimal behavior. Thus, learning a proper reward function from example behavior can introduce robustness and lower the probability of unintended behavior. The reward function captures the problem as a whole and oftentimes can generalize, so it is the ideal form to capture behavior instead of e.g. copying a policy. From the reward function, the optimal policy is completely specified, as the agent can use any one of many reinforcement learning algorithms, such as policy iteration or policy gradients, to optimize its actions based on the reward. In such cases where the exact reward function is unspecified, inverse reinforcement learning is key as it allows the agent to learn from human-provided demonstrations and form its own understanding of the intended reward function, thus enabling it to still optimize its actions and obtain a policy.

Bayesian Inverse Reinforcement Learning in particular is useful because it allows the agent to incorporate priors about the environment that the human demonstrator may have based on domain knowledge, and it models the true reward function as a probability distribution which can better capture uncertainties or noise in the demonstration data. However, there currently exist many different implementations of BIRL and it is not always clear when each one is most preferred. As such, we want to further investigate this problem and help work towards forming a unifying framework for which BIRL method is best given different conditions, such that humans and roboticists deploying AI systems in the real world can rest assured these systems are safe, accurate, and efficient. We will focus on three particular methods: the original BIRL method, Bayesian Reward EXtrapolation, and Approximate Variational Reward Imitation Learning.

We will compare each BIRL approach along several axes–required environment conditions and assumptions, sample complexity, and accuracy of the reward function estimate upon convergence—and discuss benefits and drawbacks between each approach. Our aim is that this paper can help determine which style of BIRL is preferred for which IRL applications and potentially point out future research directions for effective, efficient, and safe AI deployments.

2 Preliminaries

2.1 Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) is the process of inferring the underlying reward function of an environment or intended reward function of a demonstrator R^* using observed behaviors, usually in the form of demonstrations of a policy. These demonstrations take the form of trajectories, which are sequences of states and actions $((s_1, a_1), (s_2, a_2), ...)$. Using these and often some standard base assumptions, sometimes including that the policy comes from an agent seeking to maximize average reward or a q function, the environment can be modeled as a Markov Decision Problem, and the policy is invariant with regards to time, a probability of a reward function per state can be extracted. Note that often IRL problems assume knowledge of the transition dynamics of the MDP, which is not guaranteed in all applicable problems.

In this way, the human is able to avoid manually specifying a reward function for the AI agent. While some environments are simple enough to be able to design a reward function that can elicit the desired behavior from an agent, many others require a lot of trial-and-error and balancing of often-conflicting behavior attributes, which limits the applicability of reinforcement learning. Because the specification of reward functions can often lead to unexpected and unwanted behaviors to optimize for the reward, IRL finding a reward function to match demonstrated behaviors can often be more appropriate. Through IRL, this applicability can be broaded greatly and extended to many domains.

2.2 Markov Decision Processes

This section begins with discussing some general terminology for Markov decision processes, which form a framework for many reinforcement learning problems. In inverse and standard reinforcement learning, a Markov Decision Process (MDP) consists of a set of states S, a set of actions A, the transition function representing the probability of transitioning between states $T : S \times A \times S \to [0, 1]$, a reward function $R : S \to \mathbb{R}$, an initial state distribution S_0, and a discount factor $\gamma \in [0, 1)$. The process generally evolves over time, and such S_0 represents the distribution at time 0.

A policy π is a mapping from states to a probability distribution over actions. The expected return of a policy π under the reward function R and initial sample distribution S_0 is denoted as $V_R^\pi = \mathbb{E}_{s\sim S_0} V_R^\pi(s)$, where the value function for a state is the expected return under an infinite horizon process beginning at that state, $V_R^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma_t R(s_t)|s_0 = s]$. The Q-value function for a state-action pair is more specific and does this for a given action, and is defined as $Q_R^\pi(s, a) = R(s) + \gamma \sum_{s'\in S} T(s, a, s) V_R^\pi(s')$. Note that the value function is the expectation of the Q-value function under the distribution of actions under the state. Following prior work [1, 19, 6, 14, 2], we assume that the reward function R can be defined in terms of a linear combination of the MDP features: for an MDP with features $\phi(s) \in \mathbb{R}^k$, $R(s) = w^T \phi(s)$ where $w \in \mathbb{R}^k$ is a vector of feature weights, with $\|w\|_2 = 1$.

2.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo methods are a type of algorithm that are used to sample from complex probability distributions. For instance, discrete sample spaces with exponentially high number of values, or sample spaces in high dimensions, can be difficult to represent in a traditional algorithm and thus sample from. The Markov Chain Monte Carlo class of algorithms allow for sampling for these kinds of distributions by relying on local changes through a Markov Chain, and by running the Markov process for a number of steps, the distribution of the Markov process approaches the distribution we want to sample from in total variation distance. The difference in Markov Chain

Monte Carlo algorithms then often represents a difference in the design of the Markov chain and the transition probabilities, in order to achieve faster mixing times. Markov chain theory is often very deep in theory as well, and many tools have been created and used for determining theoretical mixing time guarantees.

Markov Chain Monte Carlo algorithms are used to sample from arbitrary distribution often make use of local structure within the distribution. For instance, a commonly used Markov Chain Monte Carlo algorithm is the Metropolis Hastings algorithm, which many other algorithms can be reduced to. The Metropolis Hastings algorithm can be described as follows: For the first sample, pick an initial state. The next state is generated as a sample from the conditional distribution of the current state. Then with certain probability, either accept or reject taking the new state. Markov Chain Monte Carlo algorithms enjoy wide application in statistical physics, where they can be used to sample from a variety of models.

2.4 Evidence Lower BOund (ELBO)

Variational methods such as the aforementioned AVRIL often require finding approximations to a distribution. Many of these methods rely on a fact called the Evidence Lower Bound, or ELBO. Besides providing a bound on the probability of a random variable based on latent variables, expansion of this provides a method for a neural network based optimization of finding a distribution.

In context of AVRIL, we let x represent the trajectory from executing a policy, and z represent the reward function inducing this policy, which is considered a latent variable. We write the ELBO as $\log p(x) \geq E_{q_\phi(z|x)}[\frac{p(x,z)}{q_\phi(z|x)}]$. The derivation is as follows:

$$\log p(x) = \log p(x) \int q_\phi(z|x) dz \tag{1}$$

$$= E_{q_\phi(z|x)}[\log p(x)] \tag{2}$$

$$= E_{q_\phi(z|x)}[\log \frac{p(x,z) q_\phi(z|x)}{p(z|x) q_\phi(z|x)}] \tag{3}$$

$$= E_{q_\phi(z|x)}[\log \frac{p(x,z)}{q_\phi(z|x)}] + E_{q_\phi(z|x)}[\log \frac{q_\phi(z|x)}{p(z|x)}] \tag{4}$$

$$= E_{q_\phi(z|x)}[\log \frac{p(x,z)}{q_\phi(z|x)}] + D_{\text{KL}}(q_\phi(z|x)||p(z|x)) \tag{5}$$

$$\geq E_{q_\phi(z|x)}[\log \frac{p(x,z)}{q_\phi(z|x)}] \tag{6}$$

where the last line follows from the KL divergence always being positive. Since the probability of the trajectory is constant if the rewards are marginalized out, increasing the ELBO term decreases the KL divergence between $q_\phi(z|x)||p(z|x)$. Remembering x represents the trajectory and z given the rewards, this means that maximizing the ELBO makes the approximator $q_\phi(z|x)$ closer to the true posterior distribution. The direct application of this to AVRIL will be shown in the AVRIL section.

3 Bayesian Inverse Reinforcement Learning

In inverse reinforcement learning (IRL), we seek the underlying reward function of an MDP given a set of human demonstrations [17]. We denote a set of demonstrations by D, where a single demonstration is defined to be a set of state-action pairs: $D = \{(s_1, a_1), \ldots, (s_n, a_n)\}$. Bayesian IRL (BIRL) [18] estimates the posterior over possible reward functions given the set of demonstrations: $P(R|D) \propto P(D|R)P(R)$, where the demonstrator is assumed to be follow a softmax policy, leading to the following likelihood function:

$$P(D|R) = \prod_{(s,a)\in D} P((s,a)|R) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}} \tag{7}$$

where $\beta \in [0, \infty)$ represents the confidence in the demonstrator's optimality (a higher β means the demonstrator is more likely to give optimal demonstrations) and $Q^*_R(s,a) = \max_\pi Q^\pi_R(s,a)$ is the

optimal Q-value for a state and action under the reward function R. Equation (7) assigns higher likelihoods to demonstrated actions that result in higher Q-values under R compared to alternative actions. Equation (7) is an example of Boltzmann-rationality, a model that has found widespread utility in economics [5, 16], psychology [3, 11, 12], and AI [20, 9, 4, 10, 13, 15] as a useful model of human decision-making and can be seen as the maximum entropy distribution over choices for a satisficing agent [13]. This means we can write $P(R|D) = \frac{1}{Z'} e^{\beta \sum_i Q^*(s_i,a_i,R)} P(R)$, where Z' is a normalizing constant consisting of $P(D)$ and the well known partition function as the above is Boltzmann distribution like.

Thus, the posterior distribution of the reward given the demonstrations has a known distribution that can be calculated. Because the distribution is written as $\frac{1}{Z'} e^{\beta \sum_i Q^*(s_i,a_i,R)} P(R)$, we can sample a reward from this complex distribution without explicitly calculating Z' which can be quite large and cumbersome. This makes use of many MCMC algorithms' property that only the ratio between probabilities of rewards needs to be known, and not the exact probability. This desired property allows BIRL to sample from the reward posterior distribution and use these samples to find the mean reward. The mean reward reduces loss across many metrics, and so is often an optimal reward to search for. However even with MCMC, the transition probabilities remain nontrivial as it requires solving the MDP after each reward sample in order to find optimal Q^* values for every state and action pair in the demonstrations, which is often extremely costly. The original BIRL paper addresses this and requires approximations of the optimal Q^* values to speed this up, but this remains a very costly method. Subsequent BIRL related methods including B-REX and AVRIL improve upon this.

The specific MCMC algorithm the original BIRL method uses is a variant of gridwalk called Policy Walk and is listed here. Note that PolicyIteration is an algorithm used for keeping track of the optimal policy for a given reward π which is not included.

Algorithm 1 Policy Walk

Require: Distribution P, MDP M, Step Size δ

1. Pick a random reward vector $R \in \mathbf{R}^{|S|}/\delta$
2. $\pi :=$ PolicyIteration(M, R)
3. Repeat
   (a) Pick a reward vector $\bar{R}$ uniformly at random from the neighbors of R in $\mathbf{R}^{|S|}/\delta$
   (b) Compute $Q^\pi(s, a, \bar{R})$ for all $(s, a) \in S, A$
   (c) If $\exists (s, a) \in Q^\pi(s, \pi(s), \bar{R}) < Q^\pi(s, a, \bar{R})$
      i. $\bar{\pi}$:= PolicyIteration$(M, \bar{R}, \pi)$
      ii. Set $R := \bar{R}$ and $\pi := \bar{\pi}$ with probability $\min\{1, \frac{P(\bar{R},\bar{\pi}}{P(R,\pi)}\}$
   Else
      i. Set $R := \bar{R}$ with probability $\min\{1, \frac{P(\bar{R},\bar{\pi}}{P(R,\pi)}\}$

4 Bayesian Reward Extrapolation (B-REX)

Next, we assess Bayesian Reward Extrapolation (B-REX) [7], which is a BIRL algorithm that can scale to complex control problems with visual observations and ensures safety in imitation learning. Its key novelty is that it greatly increases sampling efficiency from the posterior distribution by using pairwise preferences between demonstrations, $\mathcal{P} = \{(i, j) : \tau_i \prec \tau_j\}$ where $\tau_i \in D \, \forall i$. This is notable as it no longer needs an assumption of how optimal the policy is, nor have it controlled by the previous hyperparameter β. While in the original MCMC implementation, the likelihood of each reward proposal R is found by assessing how likely the provided demonstrations D are if $R^* = R$, B-REX calculates the proposal likelihoods by assessing how likely the preferences are if $R^* = R$. As such, instead of having to find Q-values for every proposal, B-REX trains a reward network R_θ once, freezes all weight layers except the last, and then uses the Bradley-Terry model (see below) to calculate the likelihoods, which is much more efficient. Furthermore, the likelihood function does not require solving an MDP or access to the MDP, only that the total reward of a trajectory can be

calculated under R_θ, which is a much less strict assumption.

$$P(\mathcal{P}, D \mid R_\theta) = \prod_{(i,j)\in\mathcal{P}} \frac{e^{\beta R_\theta(\tau_j)}}{e^{\beta R_\theta(\tau_i)} + e^{\beta R_\theta(\tau_j)}} \tag{8}$$

Furthermore, this can be optimized even further by using the properties of the neural network approximator to the reward function. By using the penultimate layer for each state and caching this, then we can write $R_\theta(s) = w^T\phi(s)$, where $\phi(s)$ are the outputs of this penultimate layer. Given $R(s) = w^T\phi(s)$, the value function then becomes

$$V_R^\pi = E_\pi[\sum_{t=0}^{T} R(s_t)] = w^T E_\pi[\sum_{t=0}^{T} \phi(s_t)] \tag{9}$$

which is much simpler and easier to compute. Recall that the $R_\theta(\tau)$ in the likelihood function is the estimated return over the trajectory, which is the V_R^π. B-REX remains a MCMC variant of BIRL, as it requires use of a MCMC algorithm to sample from the distribution induced by the previous likelihood function. The algorithm as shown in the original paper is as follows:

Algorithm 2 B-REX: Bayesian Reward Extrapolation

Require: Demonstrations D, pairewise preferences P, MCMC proposal width σ, number of proposals N, deep network architecture R_θ, prior $P(w)$.
Pretrain R_θ using auxillary tasks
Freeze all but last layer w of R_θ. Let $\phi(s)$ be activations of penultimate layer of R_θ.
Precompute and cache $\Phi_r = \sum_{s\in r} \phi(s)$ for all $r \in D$.
Initialize w.
Chain[0] $\leftarrow$ w
Compute $P(P, D|w)P(w)$ using likelihood function
for $i \leftarrow 1$ to N **do**
  $\bar{w} \leftarrow N(w, \sigma)$
  Compute $P(P, D|\bar{w})P(\bar{w})$, which is given by the likelihood function
  $u \leftarrow$ Uniform$(0, 1)$
  if $u < \frac{P(P,D|\bar{w})P(\bar{w})}{P(P,D|w)P(w)}$ **then**
    Chain[i] $\leftarrow \bar{w}$
    $w \leftarrow \bar{w}$
  else
    Chain[i] $\leftarrow \bar{w}$
  end if
end for

5 Approximate Variational Reward Imitation Learning (AVRIL)

The final Bayesian IRL method we consider is Approximate Variational Reward Imitation Learning [8], which works without using MCMC sampling.

AVRIL is a Bayesian IRL algorithm that avoids having to solve the inner Q-value loop present in many MCMC-based BIRL algorithms by jointly learning an approximate variational posterior distribution over the true reward function in an autoencoder-like manner, enabling BIRL to be much more scalable to more complex, high-dimensional environments. Using the framework from the ELBO section, we can write the ELBO as
$E_{q_\phi(z|x)}[\log \frac{p(x,z)}{q_\phi(z|x)}] = E_{q_\phi(z|x)}[\log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}] = E_{q_\phi(z|x)}[\log p_\theta(x|z)] + E_{q_\phi(z|x)}[\log \frac{p(z)}{q_\phi(z|x)}]$
which now can be written as a reconstruction term minus a prior matching term

$$E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)||p(z)) \tag{10}$$

Recalling that maximizing the ELBO minimizes the KL divergence between $q_\phi(z|x)$ and $p(z|x)$, the approximated prior and true prior, and that we defined x as the trajectory and z as the latent

rewards, we see this decomposition of the ELBO is the objective that AVRIL seeks to maximize with an additional regularizer:

$$F(\phi, \theta, D) = \log \left(\frac{e^{\beta Q_\theta(s,a)}}{\sum_{b \in A} e^{\beta Q_\theta(s,b)}} \right) - D_{KL}(q_\theta(R(s,a)) \| p(R(s,a))) + \lambda \log q_\phi(Q_\theta(s,a) - \gamma Q_\theta(s', a')) \tag{11}$$

where D_{KL} is the Kullback-Leibler divergence, and q_ϕ and Q_θ are the variational and policy Q-functions, respectively.

The practical objective function is designed to balance several factors, including the human preferences as expressed by the demonstrations and the prior distribution of the potential rewards. Through modeling BIRL as an optimization problem with this objective function, AVRIL is able to efficiently learn the posterior distribution from which the underlying reward function can be estimated.

An additionally useful feature of AVRIL is that it can operate entirely offline as it does not require the agent to test reward proposals in the environment, which is a significant advancement over MCMC-based BIRL approaches. We include the AVRIL algorithm here for completeness:

Algorithm 3 Approximate Variational Reward Imitation Learning (AVRIL)

Result: Parameters ϕ of variational distribution and θ of policy Q-function
Input: D, S, A, γ, λ, learning rate η, mini-batch size b;
Initialise ϕ, θ;
while not converged **do**
  Sample D_{mini} from D;
  $F(\phi, \theta, D) = E[\frac{n}{b} F(\phi, \theta, D_{mini}]$; ▷ Monte Carlo estimate of total loss
  $(\phi', \theta') \leftarrow (\phi, \theta) + \eta \nabla_{\phi,\theta} F(\phi, \theta, D)$; ▷ Gradient step for ϕ, θ
  $\phi, \theta \leftarrow \phi', \theta'$
end while
return ϕ, θ

6 Experiments

The experiments we will run will compare the standard BIRL, B-REX, and AVRIL. We will start with a basic 8 x 8 gridworld and vary the number of features and percentage of state coverage provided in the demonstrations. A ground truth reward will be specified that is the same for each method, and the graphs of the distance between the current estimated reward and the the ground truth reward per iteration are plotted below. Demonstrations will be created using an optimal policy with regards to the ground truth reward. These demonstrations will not necessarily cover the full state space, as the percentage of the state space covered varies.

Note that the meaning of "per iteration" differs between the methods that use MCMC (BIRL and B-REX) and AVRIL. We expect that BIRL would have the best performance in this limited setting, as the main drawback for BIRL is the time it takes per iteration. The traditional BIRL method requires the inner MDP to be solved, and in this setting it is feasible to be solved quickly and accurately. The B-REX method has a function approximator for the inner MDP solver, which could be more efficient time-wise but costly in terms of accuracy. Even more so, AVRIL has function approximators for *both* the posterior and the Q values, which would most likely cost the most accuracy in achieving faster performance.

Each gridworld configuration has either 2, 4, or 8 features and a percentage of states used for demonstration as 25, 50, 75, or 100. We ran 10 replicates of each configuration. The graphs will have features on the x-axis going from 2 to 8 and percent state coverage on the y-going from 25 to 100. We will also show heatmaps representing the ground truth reward and the final reward determined at each state by each method.

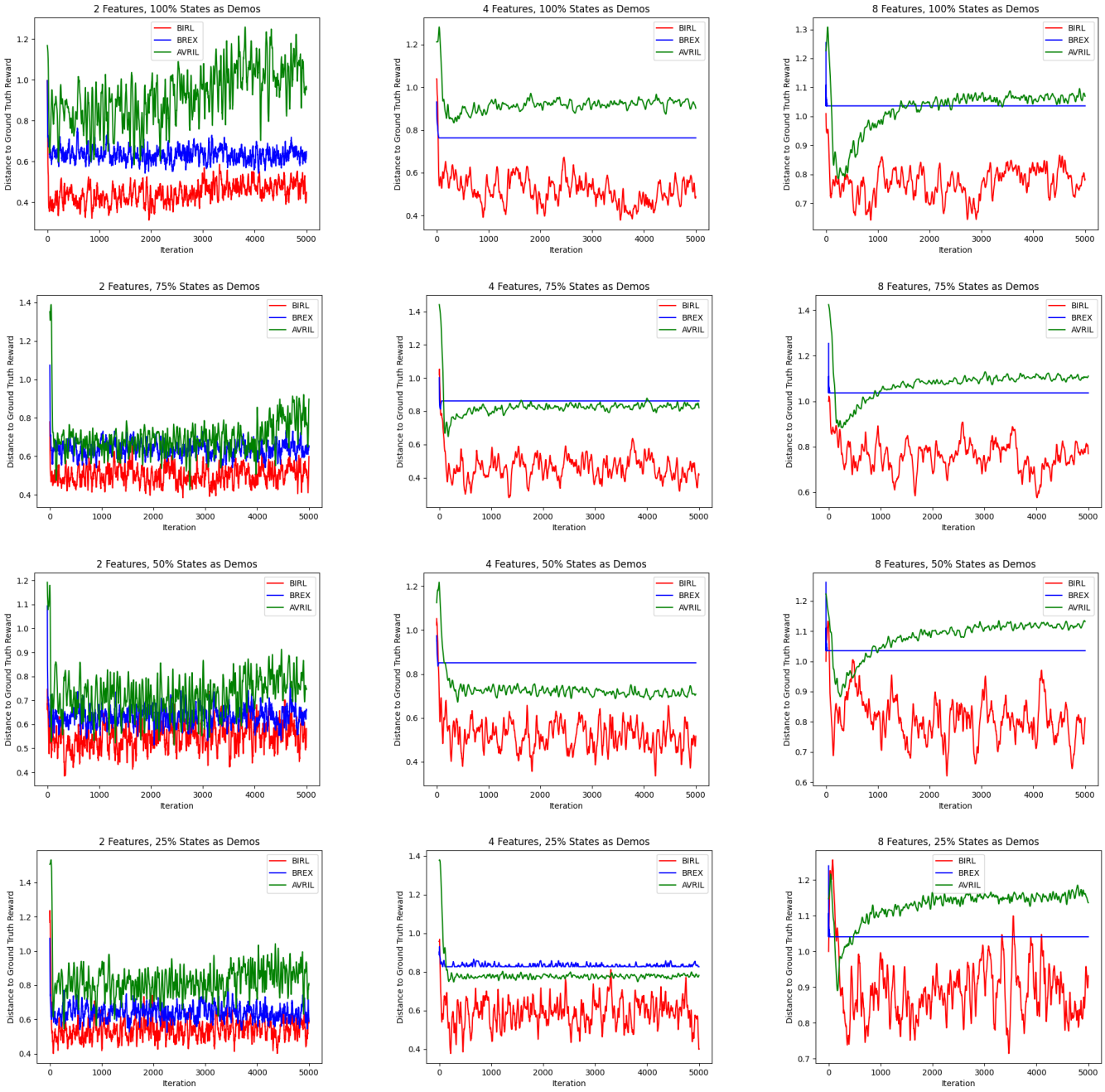

Figure 1: Plots of BIRL, B-REX and AVRIL. From left to right are features 2, 4, and 8; from bottom to top are state coverages 25%, 50%, 75% and 100%

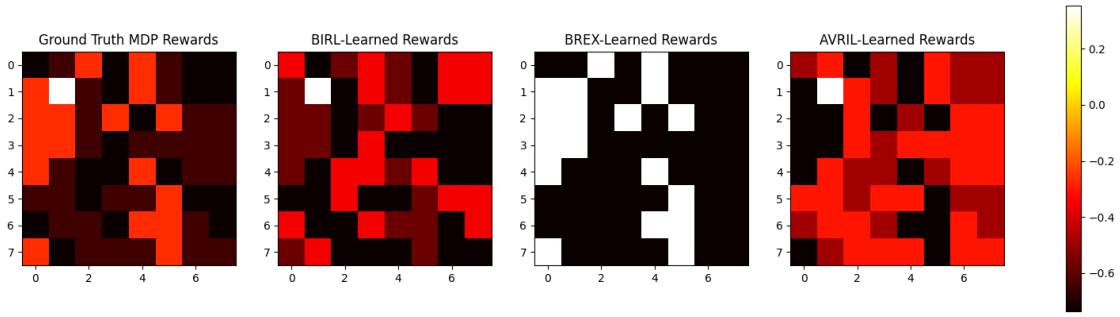

Figure 2: Reward heatmaps showing the rewards that traditional BIRL, B-REX, and AVRIL each identified

7 Results and Discussion

The results seem to show that for our experiments, BIRL performs consistently the best in terms of closer convergence to the ground-truth reward, with B-REX often performing slightly worse, and AVRIL often performing the worst. In terms of computation time, BIRL on average required 5.5 minutes to finish one environment configuration, B-REX required 1.5 minutes, and AVRIL required 1.0 minute.

We acknowledge that these are on a basic environment without much transition dynamics complexity; as such, it could be that the feasibility of solving an MDP directly each time is what gives BIRL the advantage. In other words, when the environment is simple enough, it is much better to opt for exact solving as with BIRL than using approximators as with B-REX or AVRIL, especially as the computational time difference is an acceptable sacrifice.

From the point of view of BIRL methods however, the original BIRL method represents a robust and fundamental, though costly, way of approaching the problem. The traditional BIRL requires solving the MDP completely and sampling with MCMC. B-REX advances on this by approximating the inner MDP solver with a neural network, which could come at a cost of lower accuracy but increased time and sample efficiency. AVRIL makes a further advance by using two neural network function approximators to reduce *both* the need for the inner MDP solver and the outer MCMC sampler, by using the ELBO.

Notably for the traditional BIRL method, it seems that with a higher number of features, the process has increased fluctuation. The fluctuation is the most significant for 8 features and 25% state coverage, which would make sense as each state could take on a wider range of feature weights and there is the least amount of demonstrations to guide the agent towards the correct reward function. For B-REX, the fluctuation seems to decrease as the number of features goes up, to the point where at 4 features, regardless of state coverage, seems to have almost no gradient. This is relatively surprising, as it also seems that B-REX converges to similar values conditioned on the number of features.

It also seems that as the number of features increases, the distance between the final reward proposal and the ground truth reward increases for all methods; this follows naturally as the more complex an environment is, the more difficult it will be for any method to truly concentrate on the ground truth reward given so many possible reward function values.

Considering the heatmaps, B-REX seems to be the most absolute, and only outputs two values. This seems that it would induce a 0 gradient in many places, reflecting the relatively stationary behavior at many timesteps. AVRIL also seems to converge, like B-REX, to a non-zero value. This plateau behavior is unexpected.

In the end, it is not particularly clear when to use each method. For high dimensional problems, or ones where the state and action space are particularly complicated it seems that using AVRIL will return the best, however if the transition function is known and well specified, the traditional BIRL method is still viable. Experimental testing can help one to determine which to use.

References

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 1.

[2] Saurabh Arora and Prashant Doshi. 2021. A survey of inverse reinforcement learning: challenges, methods and progress. *Artificial Intelligence*, 297, 103500.

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

[4] Michael Bloem and Nicholas Bambos. 2014. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *53rd IEEE conference on decision and control*. IEEE, 4911–4916.

[5] Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: i. the method of paired comparisons. *Biometrika*, 39, 3/4, 324–345.

[6] Daniel Brown and Scott Niekum. 2018. Efficient probabilistic performance bounds for inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* number 1. Vol. 32.

[7] Daniel S. Brown and Scott Niekum. 2019. Deep bayesian reward learning from preferences. *CoRR*, abs/1912.04472. `http://arxiv.org/abs/1912.04472` arXiv: `1912.04472`.

[8] Alex J. Chan and Mihaela van der Schaar. 2021. Scalable bayesian inverse reinforcement learning. *CoRR*, abs/2102.06483. `https://arxiv.org/abs/2102.06483` arXiv: `2102.06483`.

[9] Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. 2013. Legibility and predictability of robot motion. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 301–308.

[10] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided cost learning: deep inverse optimal control via policy optimization. In *International conference on machine learning*. PMLR, 49–58.

[11] Noah D Goodman, Chris L Baker, and Joshua B Tenenbaum. 2009. Cause and intent: social reasoning in causal learning. In *Proceedings of the 31st annual conference of the cognitive science society*. Cognitive Science Society Amsterdam, 2759–2764.

[12] Noah D Goodman and Andreas Stuhlmüller. 2013. Knowledge and implicature: modeling language understanding as social cognition. *Topics in cognitive science*, 5, 1, 173–184.

[13] Hong Jun Jeon, Smitha Milli, and Anca Dragan. 2020. Reward-rational (implicit) choice: a unifying formalism for reward learning. *Advances in Neural Information Processing Systems*, 33, 4415–4426.

[14] Ananth Jonnavittula and Dylan P Losey. 2021. I know what you meant: learning human objectives by (under) estimating their choice set. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2747–2753.

[15] Cassidy Laidlaw and Anca Dragan. 2021. The boltzmann policy distribution: accounting for systematic suboptimality in human models. In *International Conference on Learning Representations*.

[16] R Duncan Luce. 2012. *Individual choice behavior: A theoretical analysis*. Courier Corporation.

[17] Andrew Y Ng, Stuart Russell, et al. 2000. Algorithms for inverse reinforcement learning. In *ICML*. Vol. 1, 2.

[18] Deepak Ramachandran and Eyal Amir. 2007. Bayesian inverse reinforcement learning. In *IJCAI*. Vol. 7, 2586–2591.

[19] Shao Zhifei and Er Meng Joo. 2012. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*.

[20] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning. In *Aaai*. Vol. 8. Chicago, IL, USA, 1433–1438.