Active Learning from Critiques via Bayesian Inverse Reinforcement Learning

Yuchen Cui
Department of Computer Science
University of Texas at Austin, Austin, TX 78712
yuchencui@utexas.edu

Scott Niekum
Department of Computer Science
University of Texas at Austin, Austin, TX 78712
sniekum@cs.utexas.edu

***Abstract*—Learning from demonstration algorithms, such as *Inverse Reinforcement Learning*, aim to provide a natural mechanism for programming robots, but can often require a prohibitive number of demonstrations to capture important subtleties of a task. Rather than requesting additional demonstrations blindly, active learning methods leverage uncertainty to query the user for action labels at states with high expected information gain. However, this approach can still require a large number of labels to adequately reduce uncertainty and may also be unintuitive, as users are not accustomed to determining optimal actions in a single out-of-context state. To address these shortcomings, we propose a novel trajectory-based active Bayesian inverse reinforcement learning algorithm that 1) queries the user for *critiques* of automatically generated trajectories, rather than asking for demonstrations or action labels, 2) utilizes *trajectory segmentation* to expedite the critique / labeling process, and 3) predicts the user's critiques to generate the most highly informative trajectory queries. We evaluated our algorithm in a simulated domain, finding it to compare favorably to prior work and a randomized baseline.**

I. Introduction

Robots have successfully been used to automate tasks such as manufacturing, in which the environment is standardized and well-understood. However, as automation begins to expand to homes and less structured workplaces, it is not feasible for robotics engineers to program general-purpose robots to perform highly variable tasks in many different environments. In response to this challenge, recent learning from demonstration (LfD) [4] algorithms aim to provide an intuitive interface that allows end-users to program robots without the use of code or expert knowledge. *Inverse Reinforcement Learning* (IRL) is a form of LfD that focuses on recovering the underlying reward function that generates an expert's demonstrations [1]. IRL often enables generalization to situations by learning a reward function, instead of a specific policy.

A significant problem for existing IRL algorithms is that they often require a large number of demonstrations to be robust, while at the same time, it is difficult for users to determine what types of demonstrations are most informative to show the robot. *Active learning* is a framework in which the learning agent selects its own input data, leveraging uncertainty and expected information gain. The most closely related active IRL technique [13] allows a robot to query the user for a demonstration at the state with highest uncertainty over policies, which can still require substantial human effort and is unintuitive for certain types of problems where it is hard to determine the optimal action at a single out-of-context state. It is also desirable to have a method that can work when the user does not understand the action space of the robot, or is not physically present to provide additional demonstrations—for example, a robot encountering an error while deployed may contact a remote call center for assistance. Finally, the algorithm should minimize the amount of human feedback required to reduce user burden. To address these shortcomings, we propose a novel trajectory-based active Bayesian inverse reinforcement learning algorithm, ALfC-BIRL.

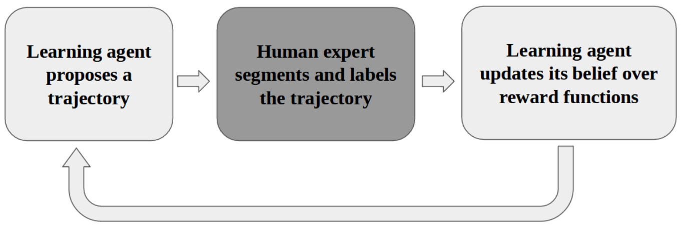

Learning agent proposes a trajectory
Human expert segments and labels the trajectory
Learning agent updates its belief over reward functions

Fig. 1: Proposed Active Learning Process

The proposed method automatically generates trajectories, which users are then asked to *critique*, rather than querying users for optimal actions or demonstrations directly. Instead of having the user to critique the trajectory as a whole, our method allows the user to segment the trajectory into *good* and *bad* sections, since trajectories are rarely purely optimal or pessimal. This approach enables users to understand actions in the context of a natural trajectory, while also allowing for the collection of many state-action labels from a small number of segmentation points. To generate trajectories that are likely to result in high information gain, we build on Bayesian Inverse Reinforcement Learning (BIRL) [15], which generates a belief distribution over possible reward functions under a given set of demonstrations. This distribution can be used to predict an expected segmentation of any given trajectory, and in turn, predict the expected change in the reward function distribution, and thus the information gain that will result from the query. While exactly finding an information-maximizing trajectory is infeasible in practice, we present an approximate algorithm for doing so, and show that it compares favorably to prior work and a random baseline in terms of policy loss, data efficiency, and required labeling effort.

II. Related Work

Inverse Reinforcement Learning (IRL), a subtype of Learning from Demonstration [4], is the process of deriving a reward function from observed behavior [1]. In contrast to approaches that aim to directly mimic the expert's behavior, such as max margin planning [16], IRL algorithms learn a reward function that describes the task and therefore is transferable to new environments. However, recovering the exact reward function is an ill-posed problem since many reward functions generate the same optimal policy, including a reward function that is zero at every state. Abbeel and Ng [1] use a max-margin algorithm to match the feature counts between the expert's policy and the learning agent's policy. Given samples from the expert's policy, there are many policies matching the feature counts. To address the ambiguity in choosing policies, Ziebart et al. [19] employed the principle of maximum entropy to find a reward function that maximizes entropy of the probability distribution over paths. However, these feature-count based techniques do not directly address the sub-optimality of demonstrations and cannot work with partial trajectories.

Bayesian Inverse Reinforcement Learning (BIRL) was first proposed by Ramachandran & Amir [15] as a principled way of approaching IRL, casting the ill-posed problem into Bayesian framework. In BIRL, demonstrated state-action pairs are each used as an independent evidence to update a posterior distribution over the space of reward functions. Choi & Kim [7] suggested to use a Maximum a Posteriori (MAP) estimation instead of taking the mean of the reward distribution as a more accurate estimate of the reward function. Our proposed method transforms BIRL into a *learning from critiques* (LfC) method and also leverages both positive and negative samples to address data-efficiency.

Lopes et al. [13] built an active learning algorithm based on BIRL that enables the robot to query the expert for demonstrations at states where the entropy over the distribution of action probabilities is high. Cohn et al. [9] proposed to use myopic expected value of information as the measure for selecting action queries instead of policy entropy. Our work is closely related to that of Lopes et al. [13] and Cohn et al. [9]. However, our algorithm reasons about information gain directly over the space of possible reward functions instead of policies or expected values. We will also demonstrate that ALfC-BIRL can be more data-efficient with the same amount of human effort since it generates a sequence of actions as its query instead of asking for individual demonstrations at out-of-context states.

There also exists work on reducing teaching burden by leveraging human feedback outside the context of IRL. Cakmak and Thomaz [5] studied how to design effective robot learners from a human-robot interaction perspective and their results in part support our design choice of using label queries instead of asking for demonstrations. Argall et al. [3] presented an approach to incorporate human critiques at policy level into an 1-Nearest-Neighbor-based LfD algorithm. In the context of Reinforcement Learning (RL), Judah et al. [10] enabled an agent to learn a parametrized policy from expert's critiques by encoding critiques into reward values. Christiano et al. [8] trained a deep network to predict rewards using feedback of human's preference over pairs of trajectory segments.

III. Background

A. Markov Decision Processes

In general, a Markov Decision Process (MDP) is a tuple $(S, A, T, R, d_0, \gamma)$, where: S is a set of states; A is a set of actions; $T : S \times A \times S \to [0, 1]$ is a transition probability function; $R : S \to \mathbb{R}$ is a reward function, with absolute value bounded by R_{max}; d_0 is a starting state distribution and $\gamma \in [0, 1)$ is the discount factor.

A deterministic policy is a mapping from state to action $\pi : S \to A$. The value of a state given a policy is calculated by:

$$V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s, \pi] \tag{1}$$

The Q-function is defined to describe values of state-action pairs according to some policy:

$$Q^{\pi}(s, a) = R(s) + \gamma \mathbb{E}_{s' \sim T(s,a,*)}[V^{\pi}(s')] \tag{2}$$

Bellman equations are used to describe a recursive relationship between values of neighboring states and state-action pairs:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') V^{\pi}(s') \tag{3}$$

$$Q^{\pi}(s, a) = R(s) + \gamma \sum_{s'} T(s, a, s') V^{\pi}(s') \tag{4}$$

A policy π is optimal if and only if:

$$\forall s \in S, \pi(s) \in \arg\max_{a \in A} Q^{\pi}(s, a) \tag{5}$$

B. Bayesian Inverse Reinforcement Learning

In the formulation of BIRL by Ramachandran & Amir [15], we consider a Markov Decision Process without reward function, denoted as MDP/R, (S, A, T, d_0, γ) and an expert χ operating in the MDP. The expert χ is assumed to be attempting to maximize the total accumulated reward according to a reward function R, using some stationary policy. The IRL agent receives a set of demonstrations $D = \{(s_0, a_0), (s_1, a_1)...(s_k, a_k)\}$. Since the policy used by χ is assumed to be stationary, we can make the independence assumption that:

$$Pr(D|R) = \prod_{i=0}^{k} Pr((s_i, a_i|R)) \tag{6}$$

According to equation (5), the reward-maximizing actions are equivalent to the actions with highest Q-values. Therefore, the likelihood of an action (s_i, a_i) given a reward function R can be modeled as:

$$Pr((s_i, a_i)|R) = \frac{1}{Z_i} e^{\alpha Q(s_i, a_i, R)} \tag{7}$$

Where α is a parameter representing the degree of confidence we have in χ's ability to choose the optimal actions [15]. Therefore, the likelihood of the entire evidence can be expressed as:

$$Pr(D|R) = \frac{1}{Z} e^{\alpha \sum_i Q(s_i, a_i, R)} \tag{8}$$

With Bayes theorem, the posterior probability of reward function R is:

$$Pr(R|D) = \frac{Pr(D|R)Pr(R)}{Pr(D)} = \frac{1}{Z'} e^{\alpha \sum_i Q(s_i, a_i, R)} Pr(R) \tag{9}$$

While the normalizing constant Z' is hard to compute, the Markov Chain Monte Carlo (MCMC) sampling algorithm only needs the ratios of probability densities. Therefore, BIRL outputs an unnormalized probability distribution of reward functions, from which we can extract a MAP estimate of the reward function R or the mean policy $\bar{\pi}$.

IV. Methodology

First, ALfC-BIRL proposes a trajectory to a human expert, who will segment the trajectory into *good* and *bad* contiguous segments. The information content will be judged by examining the expected change in the agent's belief over reward functions. The belief distribution is approximated using the MCMC sampling algorithm in BIRL. BIRL considers each state-action pair separately so it is able to learn from partial trajectories and segments, unlike feature-count based methods that require full trajectories.

A. Learning from Negative Examples using BIRL

Given that the trajectories will be segmented into *good* and *bad* parts, we modified BIRL so that it can leverage both positive and negative samples of the expert's policy. In an MDP, for a particular state s_i, an action (s_i, a_i) is either optimal or not, though multiple actions can be optimal. We assume that the expert will label the optimal actions as *good* and sub-optimal actions as *bad* but these labels may be corrupted with noise. As implied by the Bellman equations, the set of optimal actions $O(s)$ at each state s is:

$$O(s) = \arg\max_{a \in A} Q^\pi(s, a) \tag{10}$$

Following the original BIRL formulation [15], given a reward function R, the probability that the action belongs to $O(s)$ is exponentially higher if it has a larger $Q(s, a)$ value. We assume that the expert's policy is stationary and optimal under some reward function, so that demonstrations labeled as *good* all belong to $O(s)$, and those labeled as *bad* do not. Therefore, probabilities that a state-action pair is *good* or *bad* under some reward function R can be formulated as:

$$Pr(a_i \in O(s_i) \mid R) = \frac{1}{Z_i} e^{\alpha Q(s_i, a_i, R)} \tag{11}$$

$$Pr(a_i \notin O(s_i) \mid R) = 1 - \frac{1}{Z_i} e^{\alpha Q(s_i, a_i, R)} \tag{12}$$

The value of parameter α quantifies the degree of confidence or importance of a particular state-action pair (s_i, a_i) being optimal or not. This value of α can be approximated—for example, with expectation maximization [18]—but this is not the focus of this work. We denote the set of *good* trajectory segments as D^+ and the set of *bad* trajectory segments as D^-. The likelihood of the entire evidence is then expressed as:

$$Pr(D^+, D^- \mid R) = \prod_{(s_i, a_i) \in D^+} Pr(a_i \in O(s_i) \mid R) \prod_{(s_j, a_j) \in D^-} Pr(a_j \notin O(s_j) \mid R) \tag{13}$$

Algorithm 1 GenerateSamples(P, mdp, D^+,D^-,l,ϵ)

```
1: Randomly initialize reward vector $R \in \mathbb{R}^{\|S\|}$
2: $R\_chain[0] = R$
3: $\pi$ :=PolicyIteration($mdp, R$)
4: $i := 1$
5: while $i < l$ do
6:     Randomly perturb $R$ by step size $\epsilon$ and get $R'$
7:     Compute $Q^\pi(s, a, R')$ for all $(s, a) \in D^+ \cup D^-$
8:     $\pi'$ := PolicyIteration($mdp, R', \pi$)
9:     if $rand(0,1) < min\{1, \frac{P(R', \pi', D^+, D^-)}{P(R, \pi, D^+, D^-)}\}$ then
10:        $R\_chain[i] = R'$
11:        $R = R'$
12:        $i := i + 1$
13:    end if
14: end while
15: return $R\_Chain$
```

The algorithm we use to generate samples of reward functions from the posterior using D^+ and D^- is GenerateSamples as shown in Algorithm 1, which is modified from the Policy-Walk algorithm of Ramachandran & Amir [15]. It takes the likelihood function, an MDP/R, sets of positive and negative evidences, a desired chain length and a step size for modifying the reward functions as input and output an array of sampled reward functions. To reduce the autocorrelation in between the samples obtained, only every 20th sample is used in practice.

B. Calculating Expected Information Gain

In order to compare different actions in terms of their information gain, we need a measure for estimating how much information we can get by updating some state-action pair to be optimal or not. The intuition is that we want to find state-action pairs that will cause a large expected change in belief about reward functions.

It is desirable to have a measure that captures the differences in belief distributions before and after updating the optimality of a candidate state-action pair. The Kullback-Leibler (KL) divergence [12] between two distributions is widely used as a measure for information gain in information theory [2]. The equation for computing the KL divergence for two discrete distributions is:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \quad (14)$$

KL divergence is asymmetric, since $D_{KL}(P||Q)$ and $D_{KL}(Q||P)$ are not the same. While it is desired to use a symmetric measure as a distance metric, it is known that the asymmetry of KL divergence helps to avoid local optima during active learning processes [11]. The asymmetry of KL divergence will not impose a bias on candidate state-action pairs. Many other measures of distance between two probability distributions [6] may also be used here.

Since the algorithm GenerateSamples only outputs a set of samples in the MCMC chain, the KL divergence between two sets of samples is then estimated by a method based on k-Nearest-Neighbor distances [17]. However, divergence between the updated and the original distribution cannot be directly used as the expected information gain. For instance, updating some state-action pair, which is believed with 0.99 probability to be *good*, as *bad* will certainly shift the distribution by a lot, while there is little chance to actually update it to *bad*. Therefore, the algorithm also weights these distances by the probabilities of the state-action pair being optimal or not based on current belief.

Given the IRL agent's current belief of reward functions $Be(R)$, the probability of a state-action pair to be labeled as *good* or *bad* is calculated using:

$$Pr(a_i \in O(s_i) \mid Be(R)) = \sum_{R_k} Pr(R_k \mid Be(R)) Pr(a_i \in O(s_i) \mid R_k) \quad (15)$$

$$Pr(a_i \notin O(s_i) \mid Be(R)) = \sum_{R_k} Pr(R_k \mid Be(R)) Pr(a_i \notin O(s_i) \mid R_k) \quad (16)$$

The expected information gain by updating one state-action pair to be *good* or *bad* is then expressed as:

$$\begin{aligned} G^+(s_i, a_i) &= G(D^+ \cup (s_i, a_i) \mid Be(R)) \\ &= Pr(a_i \in O(s_i) \mid Be(R)) D(Be'(R) || Be(R)) \quad (17) \end{aligned}$$

$$\begin{aligned} G^-(s_i, a_i) &= G(D^- \cup (s_i, a_i) \mid Be(R)) \\ &= Pr(a_i \notin O(s_i) \mid Be(R)) D(Be'(R) || Be(R)) \quad (18) \end{aligned}$$

As presented in Algorithm 2 GetInfoGain [1], the information gain of a specific state-action pair is calculated as the sum of the weighted KL divergences.

Figure 2(a) shows an example of 5×5 gridworld with a simple layout of rewards . Figures 2(b)(c)(d)(e) present the recovered mean rewards from demonstrations and the corresponding action with the maximized expected information gain in 4 consecutive iterations running the algorithm. The selected actions tend to explore a variety of states with different

[1]In our implementation, samples for the base distribution is only obtained once before calling the GetInfoGain function for a specific state-action pair.

Algorithm 2 GetInfoGain((s,a), P, mdp, D^+, D^-, l, ϵ)

```
1:  $D^+_{tmp} := D^+ \cup (s,a)$
2:  $D^-_{tmp} := D^- \cup (s,a)$
3:  $Rwd :=$ GenerateSamples$(P, mdp, D^+, D^-, l, \epsilon)$
4:  $Rwd^+ :=$ GenerateSamples$(P, mdp, D^+_{tmp}, D^-, l, \epsilon)$
5:  $Rwd^- :=$ GenerateSamples$(P, mdp, D^+, D^-_{tmp}, l, \epsilon)$
6:  $Rwd\_total := Rwd \cup Rwd^+ \cup Rwd^-$
7:  Initialize belief arrays $Be, Be_+, Be_-$
8:  Initialize probabilities $P^+ := 0$, $P^- := 0$
9:  for each $r \in Rwd\_total$ do
10:     $\pi :=$PolicyIteration$(mdp, r)$
11:     Compute $Q^\pi(s,a,r)$ for all $(s,a) \in D^+ \cup D^-$
12:     $Be(r) := P(r, \pi, D^+, D^-)$
13:     $Be^+(r) := P(r, \pi, D^+_{tmp}, D^-)$
14:     $Be^-(r) := P(r, \pi, D^+, D^-_{tmp})$
15:     $Z^+ := 0$
16:     for $a_i \in A$ do
17:         $Z^+ := Z^+ + e^{\alpha Q(s,a_i)}$
18:     end for
19:     $P^+ := P^+ + Be^+(r)\frac{e^{\alpha Q^\pi(s,a,r)}}{Z^+}$
20:     $P^- := P^- + Be^-(r)(1 - \frac{e^{\alpha Q^\pi(s,a,r)}}{Z^+})$
21: end for
22: Normalize $Be, Be_+, Be_-$
23: $Gain^+ = D(Be_+ || Be) P_+$
24: $Gain^- = D(Be_- || Be) P_-$
25: $Gain\_total = Gain^+ + Gain^-$
26: return $Gain\_total$
```

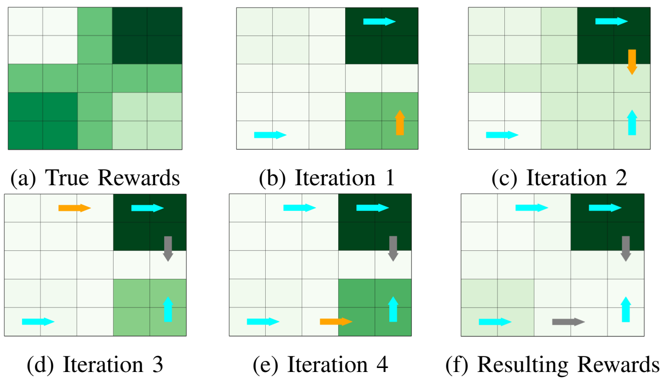


Fig. 2: Example of finding actions with maximum expected information gain in a 5×5 gridworld. (green: rewards - the darker the higher; cyan: good actions; gray: bad actions; orange: actions with max info gain.)

Iteration	Expected Information Gain	Entropy	Policy Loss
0	-	-	60%
1	4.2753338603	231.58	32%
2	4.2614594772	159.88	28%
3	4.9553412646	151.70	24%
4	5.2887902710	150.42	0%

TABLE I: Info Gain for Action Queries in Fig. 2

rewards, which agrees with our intuition for selecting informative actions. Table I shows the entropy of the distribution is decreasing over iterations and the policy loss inferred by the mean rewards also decreases over iterations.

C. Generating Informative Trajectories

By leveraging the above technique for computing the information gain of single state-action pairs, trajectories of length N can be constructed in various ways. To find the trajectory with the maximum expected information gain among all trajectories of length N starting at some state-action (s_0, a_0), we need to evaluate $O(A^N)$ trajectories, where A is the number of possible actions at every time step, which will quickly become intractable as the dimensionality of problem space increases.

Besides, trajectories generated greedily without any constraint can be arbitrarily shaped and unnatrual for humans to effectively evaluate. Therefore, instead of generating a trajectory action by action, we sample candidate trajecotries from the optimal policies of sampled reward functions. By doing so, the generated trajectories each explores a certain structure of feature weights that are consistent among states, so that they will be more interpretable to a human in terms of features. The total information gain of each of these trajectories is estimated by assuming prior actions with their expected labels and the trajectory with maximum total information gain is selected as the query to the expert. In order to estimate the information gain of a state-action pair that is not the first one in the trajectory, all prior state-action pairs are added to the demonstration sets with their expected label given current belief. Given a trajectory p of length k, its total information gain is estimated using:

$$G(p) = \sum_{(s_i, a_i) \in p} \beta^i G(s_i, a_i) \tag{19}$$

Where $\beta \in [0, 1)$ is the discount factor for a bias toward higher information gain at the beginning of the trajectory since the later state-action pairs' information gains are estimated with more and more assumptions.

V. Experiment Setup

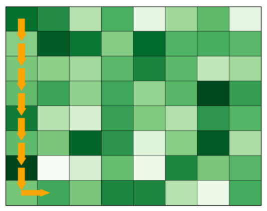
We setup our experiments in 5×5 and 8×8 gridworlds with different number of features. For ground-truth rewards, each grid cell is randomly assigned a feature vector with binary values that indicate which features are present in this cell. The reward is calculated as a linear combination of features, as assumed by prior work [1], of which the weights are randomly generated as well. The sizes of the feature vector used are 8, 16 and 32. The MDP/R problem we formulated is (S, A, T, d_0, γ) with states $||S|| = 25$ or 64 (each cell is a unique state), actions $A = \{Up, Down, Left, Right\}$, T is a deterministic transition matrix, and γ is set to be 0.95 favoring potential future rewards.

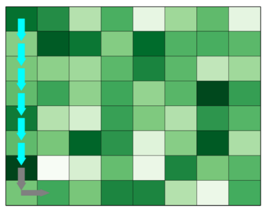
We used the optimal policy of the ground-truth rewards as the expert segmenting and labeling the generated trajectories. The convergence rate of BIRL is very sensitive to the confidence factor α as in equations (7)(8) and (9). Since we are using the optimal oracle, we set α to be relatively high at 200 so that the BIRL processes with full information for 5×5 and 8×8 gridworld converge before 4000 and 12000 samples respectively. In the cases where noise is introduced, we can adjust the value of α for each state-action pair by measuring its consistency with other demonstrations.

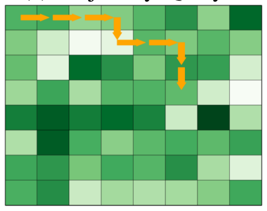
To test the performance of the algorithm, we compare ALfC-BIRL with a baseline uniform sampling algorithm and an active learning algorithm that is based on entropy over policies by Lopes et al. [13]. The entropy-based algorithm of Lopes et al. [13] is designed for LfD purpose and it is not able to generate trajectories. Therefore, in our experiments, for each iteration of the entropy-based algorithm, instead of generating a trajectory of length N, it selects N states with maximum entropies and ask the expert for demonstrations at those states.

In most practical domains, there are usually more sub-optimal actions than optimal ones in any given state, therefore, demonstrations, or positive labels, provide more valuable information than negative labels. Hence, our hypothesis is that ALfC-BIRL outperforms uniform sampling but not the entropy-based algorithm under LfD. We ran experiments using randomly generated gridworlds on each of the following algorithms:

- *ALfC-BIRL*: trajectory-based ALfC-BIRL
- *randomTraj*: basline uniform sampling algorithm
- *entropyDemo*: policy entropy-based prior work


(a) Trajectory Query 1


(b) Labeled Query 1


(c) Trajectory Query 2

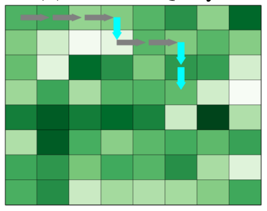

(d) Labeled Query 2

Fig. 3: Example of two trajectory queries and their feedback in 8×8 gridworlds with randomly generated features (green: true rewards; cyan: *good* actions; grey: *bad* actions; orange: trajectory query)

VI. Simulation Results

To better illustrate our approach, we present samples of trajectory queries generated by ALfC-BIRL and their ground truth segmentation points and labels to show how much effort can be saved. In order to compare the performance of ALfC-BIRL to alternative algorithms under various settings, we

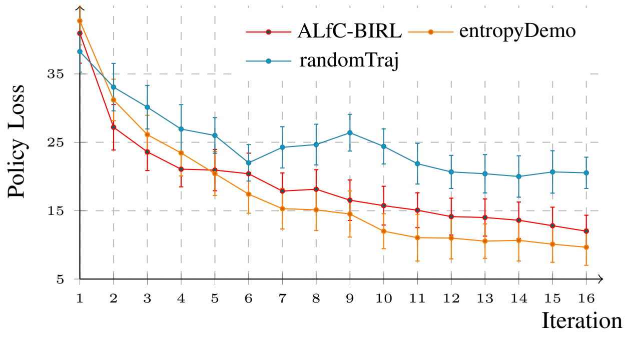


Fig. 4: Averaged Policy Loss in 40 different 5×5 gridworlds with 8 features and trajectories/queries of size 3

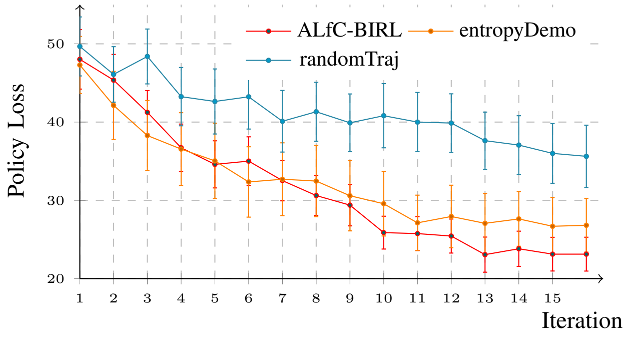


Fig. 5: Averaged Policy Loss in 40 different 8×8 gridworlds with 32 features and trajectories/queries of size 8

show the resulted average policy losses over iterations of interaction from 40 different runs of experiments per gridworld configuration.

A. *Qualitative Analysis*

Figure 3 shows two examples of generated trajectory queries from ALfC-BIRL in 8×8 gridworlds and the corresponding true labels for the generated trajectories. For these queries, the user only needs to provide a few segmentation points, one and three respectively, and the algorithm obtains eight labels in total per iteration.

In practice, the task domains normally have more structured layout than randomly generated gridworlds, so we can expect that trajectories generated by the algorithm won't consist of many small fragments. In the cases where we have to address the issue of fragmentation, we can introduce a penalty term when computing expected information gain, where an action with a different expected label than the previous k actions will produce a cost.

B. *Performance Evaluation*

Figure 4 presents the performance comparison among the three algorithms in relatively simple domains, where all three algorithms have similar performances with overlapping 95% confidence bars at early iterations but both ALfC-BIRL and the entropy-based algorithm outperform uniform sampling after six iterations. Figure 5 shows the performances of the three algorithms in more complex worlds, in which ALfC-BIRL outperforms uniform sampling with non-overlapping 95% confidence bars at most data points. It also shows that ALfC-BIRL outperforms the entropy-based algorithm in the more complex domain after about 7 or 8 iterations, which is potentially because that the more accurate the current belief model is, the more accurate the expected information gains are predicted.

VII. Conclusion

In this paper, we presented the ALfC-BIRL algorithm and discussed the major advantages of our proposed method comparing to prior work, including data efficiency, reducing human effort and enabling remote learning. ALfC-BIRL uses Bayesian inverse reinforcement learning to intelligently generate trajectories with maximum information gain, asks a user to segment the trajectory in to *good* and *bad* fragments and leverages these labeled state-action pairs to update its belief over reward functions.

Simulations have shown that ALfC-BIRL can learn policies efficiently and effectively by predicting expected information gain of its belief over reward functions. We analyzed the results qualitatively and quantitatively. Our simulated results imply that directly reasoning with the belief over reward functions is a better measure of information gain than reasoning about entropy over policies. It is also shown in simulation that using our trajectory-based active learning algorithm, an agent can learn more efficiently than uniform sampling and even outperforms the alternative algorithm with more accurate initial models in relatively complex domains.

For future work, we plan to test how the algorithm performs with noise in the expert's behavior by adjusting the confidence factor α online. We also plan to implement ALfC-BIRL on a real robotic platform to test how the algorithm's performance scales to real-world learning tasks. One major limitation of applying our proposed method to real world tasks is that we do not have a good model of human's capability giving feedback on robotic tasks, especially for segmenting and labeling trajectories, therefore we wish to obtain such model by conducting an in-depth user study. The computational cost of ALfC-BIRL is another practical concern due to the fact that we are running two MCMC processes per state-action pair in the MDP. We are exploring different methods to overcome the problem, such as implementing more efficient MCMC algorithms using Hamiltonian dynamics [14].

Acknowledgments

This work has taken place in the Personal Autonomous Robotics Lab (PeARL) at The University of Texas at Austin. PeARL research is supported in part by NSF (IIS-1638107, IIS-1617639).

References

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

[2] Yuichiro Anzai. *Pattern recognition and machine learning*. Elsevier, 2012.

[3] Brenna Argall, Brett Browning, and Manuela Veloso. Learning by demonstration with critique from a human teacher. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, pages 57–64. IEEE, 2007.

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

[5] Maya Cakmak and Andrea L Thomaz. Designing robot learners that ask good questions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 17–24. ACM, 2012.

[6] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[7] Jaedeug Choi and Kee-Eung Kim. Map inference for bayesian inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1989–1997, 2011.

[8] Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

[9] Robert Cohn, Edmund Durfee, and Satinder Singh. Comparing action-query strategies in semi-autonomous agents. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1287–1288. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[10] Kshitij Judah, Saikat Roy, Alan Fern, and Thomas G Dietterich. Reinforcement learning via practice and critique advice. In *AAAI*, 2010.

[11] Johannes Kulick, Robert Lieck, and Marc Toussaint. The advantage of cross entropy over entropy in iterative information gathering. *arXiv preprint arXiv:1409.7552*, 2014.

[12] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[13] Manuel Lopes, Francisco Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 31–46. Springer, 2009.

[14] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

[15] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51(61801):1–4, 2007.

[16] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

[17] Qing Wang, Sanjeev R Kulkarni, and Sergio Verdú. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405, 2009.

[18] Jiangchuan Zheng, Siyuan Liu, and Lionel M Ni. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *AAAI*, pages 2198–2205, 2014.

[19] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.