Deep Bayesian Reward Learning from Preferences

Daniel S. Brown
Department of Computer Science
University of Texas at Austin
dsbrown@cs.utexas.edu

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

Abstract

Bayesian inverse reinforcement learning (IRL) methods are ideal for safe imitation learning, as they allow a learning agent to reason about reward uncertainty and the safety of a learned policy. However, Bayesian IRL is computationally intractable for high-dimensional problems because each sample from the posterior requires solving an entire Markov Decision Process (MDP). While there exist non-Bayesian deep IRL methods, these methods typically infer point estimates of reward functions, precluding rigorous safety and uncertainty analysis. We propose Bayesian Reward Extrapolation (B-REX), a highly efficient, preference-based Bayesian reward learning algorithm that scales to high-dimensional, visual control tasks. Our approach uses successor feature representations and preferences over demonstrations to efficiently generate samples from the posterior distribution over the demonstrator's reward function without requiring an MDP solver. Using samples from the posterior, we demonstrate how to calculate high-confidence bounds on policy performance in the imitation learning setting, in which the ground-truth reward function is unknown. We evaluate our proposed approach on the task of learning to play Atari games via imitation learning from pixel inputs, with no access to the game score. We demonstrate that B-REX learns imitation policies that are competitive with a state-of-the-art deep imitation learning method that only learns a point estimate of the reward function. Furthermore, we demonstrate that samples from the posterior generated via B-REX can be used to compute high-confidence performance bounds for a variety of evaluation policies. We show that high-confidence performance bounds are useful for accurately ranking different evaluation policies when the reward function is unknown. We also demonstrate that high-confidence performance bounds may be useful for detecting reward hacking.

1 Introduction

As robots and other autonomous agents enter our homes, schools, workplaces, and hospitals, it is important that these agents can safely learn from and adapt to a variety of human preferences and goals. One common way to learn preferences and goals is via imitation learning, in which an autonomous agent learns how to perform a task by observing demonstrations of the task. While there exists a large body of literature on high-confidence off-policy evaluation in the reinforcement learning (RL) setting, there has been much less work on high-confidence policy evaluation in the imitation learning setting where reward samples are unavailable.

Prior work on high-confidence policy evaluation for imitation learning has used Bayesian inverse reinforcement learning (IRL) [37] to allow an agent to reason about reward uncertainty and policy robustness [13, 10]. However, Bayesian IRL is typically intractable for complex problems due to the need to repeatedly solve an MDP in the inner loop. This high computational cost precludes robust safety and uncertainty analysis for imitation learning in complex high-dimensional problems.

We first formalize the problem of high-confidence off-policy evaluation [44] for imitation learning [3]. We next propose a novel algorithm, Bayesian Reward Extrapolation (B-REX), that uses a pairwise ranking likelihood to significantly reduce the computational complexity of generating samples from the posterior distribution over reward functions when performing Bayesian IRL. We demonstrate that B-REX can leverage neural network function approximation and successor features [6] to efficiently perform deep Bayesian reward inference given preferences over demonstrations that consist of raw visual observations. Finally, we demonstrate that samples obtained from B-REX can be used to solve the high-confidence off-policy evaluation problem for imitation learning in high-dimensional tasks. We evaluate our method on imitation learning for Atari games and demonstrate that we can efficiently compute high-confidence bounds on the worst-case performance of a policy and that these bounds are beneficial when comparing different evaluation policies and may provide a useful tool for detecting reward hacking [2].

2   Related work

Imitation learning is the problem of learning a policy from demonstrations of desired behavior. Imitation learning can roughly be divided into techniques that use behavioral cloning and techniques that use inverse reinforcement learning. Behavioral cloning methods [36] seek to solve the imitation learning problem via supervised learning where the goal is to learn a mapping from states to actions that mimics the demonstrator. While computationally efficient, these methods can suffer from compounding errors [38]. Methods such as DAgger [38] and DART [29] avoid this problem by collecting additional state-action pairs from a demonstrator in an online fashion.

Inverse reinforcement learning methods [33] typically seek to solve the imitation learning problem by first estimating a reward function that makes the demonstrations appear near optimal and then performing reinforcement learning [40] on the inferred reward function to learn a policy that can generalize to states not seen in the demonstrations. Classical approaches typically repeatedly alternate between reward estimation and full policy optimization. Bayesian IRL [37] generates samples from the posterior distribution over rewards, whereas other methods seek a single estimate of the reward that matches the demonstrator's state occupancy [1], often while also seeking to maximize the entropy of the resulting policy [47]. Modern, deep learning approaches to inverse reinforcement learning are typically based on a maximum entropy framework [19] or an occupancy matching framework [26] and are related to Generative Adversarial Networks [21, 18]. These methods scale to complex control problems by iterating between reward learning and policy learning steps. Recently, Brown et al. proposed to use preferences over suboptimal demonstrations to efficiently learn a reward function via supervised learning without requiring fully or partially solving an MDP [12, 11]. The reward function is then used to optimize a potentially better-than-demonstrator policy. However, despite recent successes of deep IRL, existing methods typically return a point estimate of the reward function, precluding the rich uncertainty and robustness analysis possible with a full Bayesian approach. One of our contributions is to propose the algorithm B-REX, first deep Bayesian IRL algorithm that can scale to complex control problems with visual observations.

Another contribution of this paper is an application of B-REX to safe imitation learning [13]. While there has been much recent interest and progress in imitation learning [4], less attention has been given to problems related to safe imitation learning. Zhang and Cho propose SafeDAgger [46] a variant of DAgger that predicts in which states the novice policy will have a large action difference from the expert policy. Control is given the the expert policy only if the predicted action difference of the novice is above some hand-tuned parameter, τ. Other work has focused making generative adversarial imitation learning [26] more robust and risk-sensitive. Lacotte et al. [28] propose an imitation learning algorithm that seeks to match the tail risk of the expert as well as find a policy that is indistinguishable from the demonstrations. Brown and Niekum [13] propose a Bayesian sampling approach to provide explicit high-confidence safety bounds in the imitation learning setting. Their method uses samples from the posterior distribution $P(R|D)$ to compute sample efficient probabilistic upper bounds on the policy loss of any evaluation policy. Brown et al. [10] extend this work by proposing an active learning algorithm that uses these high-confidence performance bounds for risk-aware policy improvement via active queries. Our work presented in this paper extends and generalizes the work of Brown and Niekum [13] by demonstrating, for the first time, that high-confidence performance bounds can be obtained for imitation learning problems where demonstrations consist of high-dimensional visual observations.

Safety has been extensively studied within the reinforcement learning community (see Garcia et al. [20] for a survey). These approaches usually either seek safe exploration strategies or seek to optimize an objective other than expected return. Recently, objectives based on measures of risk such as VaR and Conditional VaR have been shown to provide tractable and useful risk-sensitive measures of performance for MDPs [41, 14]. Other related work on safe reinforcement learning has focused finding robust solutions to MDPs using Bayesian ambiguity sets [35] and on obtaining high-confidence off-policy bounds on the performance of an evaluation policy [44, 24]. Recently, it has been shown that high-confidence off policy evaluation is possible when samples of the true reward are available but the behavior policy is unknown [22]. Our work complements existing work on high-confidence off policy evaluation by formulating and providing a deep learning solution to the problem of high-confidence off-policy evaluation in the imitation learning setting, where samples of rewards are *not observed* and the demonstrator's policy (the behavioral policy) is unknown.

3 Preliminaries

3.1 Notation

We model the environment as a Markov Decision Process (MDP) consisting of states $\mathcal{S}$, actions $\mathcal{A}$, transition dynamics $T : S \times A \times S \to [0,1]$, reward function $R : S \to \mathbb{R}$, initial state distribution S_0, and discount factor γ. A policy π is a mapping from states to a probability distribution over actions. We denote the value of a policy π under reward function R as $V_R^\pi = \mathbb{E}_{s_0 \sim S_0}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi]$ and denote the value of executing policy π starting at state $s \in S$ as $V_R^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi, s_0 = s]$. Given a reward function R, we denote the Q-value of a state-action pair (s,a) as $Q_R^\pi(s,a) = R(s) + \gamma \sum_{s' \in S} T(s,a,s') V_R^\pi(s')$. We use the notation $V_R^* = \max_\pi V_R^\pi$ and $Q_R^*(s,a) = \max_\pi Q_R^\pi(s,a)$.

3.2 Bayesian Inverse Reinforcement Learning

In inverse reinforcement learning, the environment is modeled as an MDP\R where the reward function R is internal to the demonstrator and is unknown and unobserved by the learner. The goal of inverse reinforcement learning (IRL) is to infer the latent reward function of the demonstrator given demonstrations consisting of state-action pairs from the demonstrator's policy. Bayesian IRL models the demonstrator as a Boltzman rational agent that follows the softmax policy

$$\pi_R(a|s) = \frac{e^{\beta Q_R^*(s,a)}}{\sum_{b \in \mathcal{A}} e^{\beta Q_R^*(s,b)}}, \tag{1}$$

where R is the reward function of the demonstrator, and $\beta \in [0, \infty)$ is the inverse temperature parameter that represents the confidence that the demonstrator is acting optimally. Given the assumption of Boltzman rationality, the likelihood of a set of demonstrations $D = \{(s,a) : (s,a) \sim \pi_D\}$, given a specific reward function hypothesis R, can be written as

$$P(D|R) = \prod_{(s,a) \in D} \pi_R(a|s) = \prod_{(s,a) \in D} \frac{e^{\beta Q_R^*(s,a)}}{\sum_{b \in \mathcal{A}} e^{\beta Q_R^*(s,b)}}. \tag{2}$$

Bayesian IRL [37] generates samples from the posterior distribution $P(R|D) \sim P(D|R)P(R)$ via Markov Chain Monte Carlo (MCMC) sampling. This requires repeatedly solving for Q_R^* in order to compute the likelihood of each new proposal. Thus, Bayesian IRL methods are typically only used for low-dimensional problems with reward functions that are often linear combinations of a small number of hand-crafted features [13, 8]. One of our contributions is to propose an efficient deep Bayesian reward learning algorithm that leverages preferences to allow Bayesian IRL to be scaled to high-dimensional visual control problems.

4 High Confidence Off-Policy Evaluation for Imitation Learning

Before detailing B-REX, we first formalize the problem of high-confidence off-policy evaluation for imitation learning. We assume an MDP\R, an evaluation policy π_{eval}, a set of demonstrations, $D = \{(s_1, a_1), \ldots, (s_m, a_m)\}$, confidence level δ, and performance statistic $g : \Pi \times \mathcal{R} \to \mathbb{R}$, where $\mathcal{R}$ denotes the space of all reward functions and Π is the space of all policies.

The *High-Confidence Off-Policy Evaluation problem for Imitation Learning* (HCOPE-IL) is to find a high-confidence lower bound $\hat{g} : \Pi \times \mathcal{D}$ such that $\Pr(g(\pi_{\text{eval}}, R^*) \geq \hat{g}(\pi_{\text{eval}}, D)) \geq 1 - \delta$, where R^* denotes the demonstrator's true reward function, and $\mathcal{D}$ denotes the space of all demonstration sets D. HCOPE-IL takes as input an evaluation policy π_{eval}, a set of demonstrations D, and a performance statistic, $g(\pi)$, which evaluates a policy under a reward function. The goal of HCOPE-IL is to return a high-confidence lower bound $\hat{g}$ on the performance statistic $g(\pi_{\text{eval}}, R^*)$.

Note that this problem setting is significantly more challenging than the standard high-confidence off-policy evaluation problem in reinforcement learning, which we denote as HCOPE-RL. In HCOPE-RL the behavior policy is typically known and the demonstrations from the behavior policy contain ground-truth reward samples [44]. In HCOPE-IL, the behavior policy is the demonstrator's policy π_{R^*}, which is unknown. Furthermore, in HCOPE-IL the demonstration data from π_{R^*} contains only state-action pairs; samples of the true reward signal are not available. In the following sections we describe how to use preferences to scale Bayesian IRL to high-dimensional visual control tasks as a way to efficiently solve the HCOPE-IL problem for complex, visual imitation learning tasks.

5 Deep Bayesian Reward Extrapolation

Prior work [13, 10] has investigated HCOPE-IL for simple problem domains where repeatedly solving for optimal Q-values is possible. However, for high-dimensional tasks such as learning control policies from pixel observations, even solving a single MDP can be challenging and sampling from $P(R|D)$ becomes intractable. We now describe one of the main contribution of this paper: scaling Bayesian IRL to high-dimensional visual imitation learning problems.

Our first insight towards solving this problem is that the main bottleneck for standard Bayesian IRL [37] is computing the softmax likelihood function:

$$P(D|R) = \prod_{(s,a) \in D} \frac{e^{\beta Q_R^*(s,a)}}{\sum_{b \in \mathcal{A}} e^{\beta Q_R^*(s,b)}}. \tag{3}$$

which requires solving for optimal Q-values. Thus, to make Bayesian IRL scale to high-dimensional visual domains, it is necessary to either efficiently solve for optimal Q-values or to formulate a new likelihood. Value-based reinforcement learning focuses on solving for optimal Q-values quickly; however, even for low-resolution visual control tasks such as Atari, RL algorithms can several hours or even days to train [32, 25]. Because MCMC is sequential in nature, evaluating large numbers of proposal steps is infeasible given the current state-of-the-art in RL. Methods such as transfer learning could reduce the time needed to calculate Q_R^* for a new proposed reward R; however, transfer learning is not guaranteed to speed up reinforcement learning on the new task [42] and transfer learning methods that avoid performing reinforcement learning only provide loose bounds on policy performance [6], making it difficult to compute accurate likelihood ratios needed for Bayesian inference [37]. Thus, we focus on reformulating the likelihood function to speed up Bayesian IRL.

An ideal likelihood function would require little computation and minimal interaction with the environment. One promising candidate is to leverage recent work on learning from ranked demonstrations [15, 12, 11]. Given ranked demonstrations, Brown et al. [12] proposed the algorithm Trajectory-ranked Reward Extrapolation (T-REX) that performs efficient reward inference by transforming reward function learning into a classification problem using a standard pairwise ranking loss. T-REX removes the need to repeatedly sample from or solve an MDP in the inner loop, allowing IRL to scale to visual imitation learning domains such as Atari. However, T-REX only solves for the maximum likelihood estimate of the reward function. One of our contributions is to show that a similar approach based on a pairwise preference likelihood can allow for efficient sampling from the posterior distribution over reward functions.

We assume that we have a set of m trajectories $D = \{\tau_1, \ldots, \tau_m\}$ along with a set of pairwise preferences over trajectories $\mathcal{P} = \{(i, j) : \tau_i \prec \tau_j\}$. Note that we do not require a total-ordering over trajectories. These preferences may come from a human demonstrator or could be automatically generated by watching a learner improve at a task [12] or via noise injection [11]. Some trajectory pairs may not have preference information and some trajectories maybe equally preferred, i.e. (i, j) and (j, i) may both be in set $\mathcal{P}$. The benefit of pairwise preferences over trajectories is that we can now leverage a pair-wise ranking loss to compute the likelihood of a $\mathcal{P}$ given a parameterized reward function hypothesis R_θ. We use the standard Bradley-Terry model [9], alternatively called the Luce's

choice axiom [30], to obtain the following pairwise ranking likelihood function:

$$P(\mathcal{P}, D \mid R_\theta) = \prod_{(i,j) \in \mathcal{P}} \frac{e^{\beta R_\theta(\tau_j)}}{e^{\beta R_\theta(\tau_i)} + e^{\beta R_\theta(\tau_j)}}, \tag{4}$$

where $R_\theta(\tau) = \sum_{s \in \tau} R_\theta(s)$ is the predicted return of trajectory τ under the reward function R_θ, and β is the inverse temperature parameter that models the confidence in the preference labels.

Note that using the likelihood function defined in Equation (4) does not require solving an MDP. In fact, it does not require any rollouts or access to the MDP. All that is required is that we first calculate the return of each trajectory under R_θ. We then compare the relative predicted returns to the preference labels to determine the likelihood of the demonstrations under the reward hypothesis R_θ. Given this preference-based likelihood function we can perform preference-based Bayesian reward learning using standard MCMC.

5.1 Optimizations via Successor Features

B-REX uses a deep network to represent the reward function R_θ. However, MCMC proposal generation and mixing time can be slow if there are many demonstrations and if the network is especially large. To make B-REX more efficient and practical, we propose to limit the proposal to only change the last layer of weights in R_θ when generating MCMC proposals—we will discuss pretraining R_θ in a later section. We freeze all but the last layer of weights and use the activations of the penultimate layer as our reward features $\phi(s)$. This allows us to represent the reward at a state as a linear combination of features $R_\theta(s) = w^T \phi(s)$. There are two advantages to this formulation: (1) the proposal dimension for MCMC is significantly reduced, allowing for faster convergence; (2) we can efficiently compute the expected value of a policy via a single dot product, and (3) the computation required to calculated the proposal likelihood is significantly reduced.

Given $R(s) = w^T \phi(s)$, the value function of a policy can be written as

$$V_R^\pi = \mathbb{E}_\pi[\sum_{t=0}^{T} R(s_t)] = \mathbb{E}_\pi[\sum_{t=0}^{T} w^T \phi(s_t)] = w^T \mathbb{E}_\pi[\sum_{t=0}^{T} \phi(s_t)] = w^T \Phi_\pi, \tag{5}$$

where we assume a finite horizon MDP with horizon T and where Φ_π are the successor features [6] of π. Given any evaluation policy π_{eval}, we can compute the successor feature once to obtain, Φ_{eval}. We can then compute the expected value of π_{eval} as $w^T \Phi_{\pi_{\text{eval}}}$ for any reward function weights, w.

Using a linear combination of features also allows us to efficiently compute the pairwise ranking losses in the likelihood function. Consider the likelihood function in Equation (4). A naive computation of the likelihood would require $O(T \cdot |\mathcal{P}|)$ forward passes through the deep neural network R_θ per proposal evaluation, where $|\mathcal{P}|$ is the number of pairwise preferences over demonstration trajectories and T is the length of the trajectories. Given that we would like to potentially generate thousands of samples from the posterior distribution over reward functions, this can significantly slow down MCMC. However, we can reduce this computational cost by noting that

$$R_\theta(\tau) = \sum_{s \in \tau} w^T \phi(s) = w^T \sum_{s \in \tau} \phi(s) = w^T \Phi_\tau. \tag{6}$$

Thus, we can precompute and cache $\Phi_{\tau_i} = \sum_{s \in \tau_i} \phi(s)$ for $i = 1, \ldots, m$. The likelihood can then be quickly evaluated as

$$P(\mathcal{P}, D \mid R_\theta) = \prod_{(i,j) \in \mathcal{P}} \frac{e^{\beta w^T \Phi_{\tau_j}}}{e^{\beta w^T \Phi_{\tau_j}} + e^{\beta w^T \Phi_{\tau_i}}}. \tag{7}$$

This results in only $O(|\mathcal{P}|)$ dot products per proposal, resulting in a significant computational savings when generating long MCMC chains over deep neural networks.

When we refer to B-REX in the remainder of this paper we will refer to the optimized version described in this section. See Algorithm 1 in the Appendix for full pseudo-code. We found that generating 100,000 reward hypothesis for Atari imitation learning tasks takes approximately 5 minutes on a Dell Inspiron 5577 personal laptop with an Intel i7-7700 processor and an NVIDIA GTX 1050 GPU. In comparison, using standard Bayesian IRL to generate *one sample* from the posterior takes 10+ hours of training for a parallelized PPO reinforcement learning agent [39, 16].

5

5.2 Pretraining the Reward Function Network

Precompute the successor features Φ_τ assumes that we already know a good $\phi(s)$. But how do we train $\phi(s)$ from raw visual features? One way is to pretrain R_θ using T-REX [12] to find the weight parameters that result in a maximum likelihood estimate given the rankings. Then we can freeze all but the last layer of weights and perform MCMC. Another option is to train the network using an auxiliary loss. Possible candidate auxiliary losses are (1) an inverse dynamics model that uses embeddings $\phi(s_t)$ and $\phi(s_{t+1})$ to predict the corresponding action a_t [45, 23], (2) a variational pixel-to-pixel autoencoder where $\phi(s)$ is the learned latent encoding [31, 17], (3) a cross-entropy loss to learn an embedding $\phi(s)$ that can be used to classify how many timesteps apart are two randomly chosen frames [5], and (4) a forward dynamics model that predicts s_{t+1} from $\phi(s_t)$ and a_t [34, 43].

5.3 HCOPE-IL via B-REX

We now discuss how to use B-REX to find solutions to the high-confidence off-policy evaluation for imitation learning (HCOPE-IL) problem (see Section 4) when learning from raw visual demonstrations. Given samples from the distribution $P(w|D, \mathcal{P})$, where $R(s) = w^T\phi(s)$, we can compute the posterior distribution over any performance statistic $g(\pi_{\text{eval}}, R^*)$ as follows. For each sampled weight vector w produced by B-REX, we compute $g(\pi_{\text{eval}}, w)$. This results in a sample from the posterior distribution $P(g(\pi_{\text{eval}}, R)|\mathcal{P}, D)$, the posterior distribution over performance statistic g conditioned on D and $\mathcal{P}$. We then compute a $(1 - \delta)$ confidence lower bound, $\hat{g}(\pi_{\text{eval}}, D)$, by finding the δ-quantile of $g(\pi_{\text{eval}}, w)$ for $w \sim P(w|\mathcal{P}, D)$. In our experiments we focus on bounding the expected value of the evaluation policy, i.e., $g(\pi_{\text{eval}}, R^*) = V_{R^*}^{\pi_{\text{eval}}} = w^{*T}\Phi_{\pi_{\text{eval}}}$. To compute a $1 - \delta$ confidence bound on $V_{R^*}^{\pi_{\text{eval}}}$, we take full advantage of the successor feature representation to efficiently calculate the posterior distribution over policy returns given preferences and demonstrations via a simple matrix vector product, $W\Phi_{\pi_{\text{eval}}}$, where each row of W is a sample, w, from the MCMC chain and π_{eval} is the evaluation policy. We then sort the elements of this vector and select the δ-quantile. This gives us a $1 - \delta$ confidence lower bound on $V_{R^*}^{\pi_{\text{eval}}}$ and corresponds to calculating the δ-Value at Risk (VaR) over $V_R^{\pi_{\text{eval}}} \sim P(R|\mathcal{P}, D)$ [13, 27, 41].

6 Experimental Results

6.1 Imitation Learning via B-REX

We first tested the efficacy of B-REX to see if it can be used to find a reward function that leads to good policies via reinforcement learning. We enforce constraints on the weight vectors by normalizing the output of the weight vector proposal such that $\|w\|_1 = 1$ and use a Gaussian proposal function centered on w with standard deviation σ. Thus, given the current sample w_t, the proposal is defined as $w_{t+1} = \texttt{normalize}(\mathcal{N}(w_t, \sigma))$, where $\texttt{normalize}$ projects the sample back to the surface of the L1-unit ball. We used models pretrained from pairwise preferences using T-REX to obtain $\phi(s)$ [12].[1] This results in a 65 dimensional features vector $\phi(s)$. We ran MCMC for 100,000 steps with $\sigma = 0.005$ and with a uniform prior. Due to our proposed optimizations this only required a few minutes of computation time. We then took the MAP and mean reward estimates and optimized a policy using Proximal Policy Optimization [39].
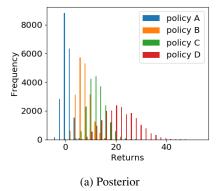
We tested our approach on five Atari games from the Arcade Learning Environment [7]. Because we are concerned with imitation learning, we mask game scores and life information and the imitation learning agent does not receive the ground-truth reward signal. All that is available are pairwise preferences on state trajectories. Table 1 shows results of performing RL on the mean and MAP rewards found using B-REX. We ran all algorithms using the same demonstrations and preference labels. We used the same 12 suboptimal demonstrations used by Brown et al. and give each algorithm all pairwise preference labels based on the ground-truth returns. T-REX uses a sigmoid to normalize rewards before passing them to the RL algorithm; however, we obtained better performance for B-REX by feeding the unnormalized predicted reward $R_\theta(s)$ into PPO for policy optimization.

Table 1 shows that, similar to T-REX, B-REX is able to utilize preferences to outperform the demonstrator. B-REX is competitive with T-REX, achieving better average scores on 3 out of 5

---

[1]Pretrained networks are available at `https://github.com/hiwonjoon/ICML2019-TREX/tree/master/atari/learned_models/icml_learned_rewards`

Table 1: Ground-truth average returns for several Atari games when optimizing the mean and MAP rewards found using B-REX. We also compare against reported results for T-REX [12]. Each algorithm is given the same 12 demonstrations with ground-truth pairwise preferences. The average performance for each IRL algorithm is the average over 30 rollouts.

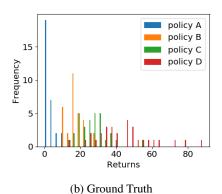|	Ranked Demonstrations		B-REX Mean	B-REX MAP	T-REX
Game	Best	Average	Average	Average	Average
Beam Rider	1332	686.0	878.7	1842.6	**3,335.7**
Breakout	32	14.5	392.5	**419.7**	221.3
Enduro	84	39.8	450.1	569.7	**586.8**
Seaquest	600	373.3	**967.3**	570.7	747.3
Space Invaders	600	332.9	1437.5	**1440.2**	1,032.5

(a) Posterior

(b) Ground Truth

Figure 1: Breakout return distributions over the posterior $P(R|D,\mathcal{P})$ compared with ground truth game scores. Policies A-D correspond to checkpoints of an RL policy partially trained on the ground-truth reward function and correspond to 25 (A), 325 (B), 800 (C), and 1450 (D) training updates to PPO. The learned posterior distributions roughly match the general shapes of the true distribution.

games. Additionally, we found that using the MAP reward from the posterior was superior to optimizing for the mean reward on 4 out of 5 games. We also found that B-REX can successfully use pairwise preferences over suboptimal demonstrations to learn a better-than-demonstrator policy. When optimizing for the mean reward, B-REX is able to obtain an average performance that surpasses the performance of the best demonstration in every game except for Beam Rider. When optimizing for the MAP reward, B-REX is obtains an average performance that surpasses the best demonstration on all games, except for Seaquest.

6.2 High-Confidence Lower Bounds on Policy Performance

Next we ran an experiment to validate whether the posterior distribution generated by B-REX can be used for accurately bounding the expected return of the evaluation policy under the unknown reward function R^*. We estimated $\Phi_{\pi_{\text{eval}}}$ using 30 Monte Carlo rollouts. We first evaluated four different evaluation policies, $A \prec B \prec C \prec D$, created by partially training a PPO agent on the ground-truth reward function. We ran B-REX to generate 100,000 samples from $P(R|\mathcal{P}, D)$. Figure 1 shows predicted and ground truth distributions for the four different evaluation policies: A–D. We found that the predicted distributions (100,000 MCMC samples) have roughly similar shape to the ground truth distribution (30 rollouts). Note we do not know the scale of the true reward R^*. Thus, the results from B-REX are most useful when comparing the relative performance of several different evaluation policies [13]. We see that the modes predicted by B-REX match the ordering of the modes of policies A–D under the true reward function.

Table 2 shows the numerical results for evaluating policies under $P(R \mid D, \mathcal{P})$. We show results for partially trained policies A-D as well as well as policies trained on the MAP reward, the mean reward, and a No-Op policy. We found that the ground-truth returns for the checkpoints were highly

Table 2: Policy evaluation statistics for Breakout over the return distribution from the learned posterior $P(R|D, \mathcal{P})$ compared with the ground truth returns using game scores. Policies A-D correspond to checkpoints of an RL policy partially trained on the ground-truth reward function and correspond to 25, 325, 800, and 1450 training updates to PPO. The mean and MAP policies are the results of PPO using the mean and MAP rewards, respectively. No-Op is a policy that never takes the action to release the ball, resulting in a lower 0.05-quantile return (0.05-VaR).

Policy	Mean Chain	0.05-VaR Chain	Traj. Length	GT Avg. Return	GT Min. Return
policy A	0.8	-1.7	213.8	2.2	0
policy B	7.4	3.6	630.1	16.6	9
policy C	12.4	7.5	834.5	26.7	12
policy D	21.5	11.6	1070.6	43.8	14
mean	88.1	25.2	3250.4	392.5	225
MAP	2030.4	75.7	29761.4	419.7	261
No-Op	6256.2	-134.9	99994.0	0.0	0

correlated with the mean reward found under the chain and the 0.05-VaR (5th percentile policy return) under the chain. However, we also noticed that the trajectory length was also highly correlated with the ground-truth reward. If the reward function learned via IRL gives a small positive reward at every timestep, then long polices that do the wrong thing may look good under the posterior. To test this we evaluated a No-Op policy that seeks to hack the learned reward function by never releasing the ball in Breakout. We ran the No-Op policy until the Atari emulator timed out after 99,994 no-ops.

The bottom row of Table 2 shows that while the No-Op policy has a high expected return over the chain, looking at the 0.05-VaR shows that the No-Op policy has high risk under the distribution, much lower than even policy A which on average only scores 2.0 points. This finding validates the results by Brown and Niekum [13] that demonstrated the value of using a probabilistic worst-case bound for evaluating the performance of policies when the true reward function is unknown. Our results demonstrate that reasoning about probabilistic worst-case performance may be one potential way to detect policies that have overfit to certain features in the demonstrations that are correlated with the intent of the demonstrations, but do not lead to desired behavior, the so-called reward hacking problem [2]. See the Appendix for results for all games. We found that for some of the games, the learned posterior is not as useful for accurately ranking policies. We hypothesize that this may be because the pretrained features $\phi(s)$ are overfit to the rankings. In the future we hope to improve these results by using additional auxiliary losses when pretraining the reward features (see Section 5.2).

7   Summary

Bayesian reasoning is a powerful tool when dealing with uncertainty and risk; however, existing Bayesian inverse reinforcement learning algorithms require solving an MDP in the inner loop, rendering them intractable for complex problems where solving an MDP may take several hours or even days. In this paper we propose a novel deep learning algorithm, Bayesian Reward Extrapolation (B-REX), that leverages preference labels over demonstrations to make Bayesian IRL tractable for high-dimensional visual imitation learning tasks. B-REX can sample tens of thousands of reward functions from the posterior in a matter of minutes using a consumer laptop. We tested our approach on five Atari imitation learning tasks and demonstrated B-REX is competitive with state-of-the-art imitation learning methods. Using the posterior samples produced by B-REX, we demonstrated for the first time that it is computationally feasible to compute high-confidence performance bounds for arbitrary evaluation policies given demonstrations of visual imitation learning tasks. Our proposed bounds can allow accurate comparison of different evaluation policies and provide a potential way to detect reward hacking. In the future we are interested in using high-confidence bounds on policy performance to implement safe and robust policy improvement in the imitation learning setting. Given a starting policy π we want to optimize a policy such that it maximizes some safety threshold. One possible way to improve the policy would be to use an evolutionary strategy where the fitness is simply the lower bound on the performance metric calculated over the posterior distribution of reward functions. We also plan to experiment with different architectures and different pretraining schemes for learning reward features automatically from raw visual features.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st international conference on Machine learning*, 2004.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[3] 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.

[4] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.

[5] Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*, 2018.

[6] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[7] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[8] Erdem Bıyık, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning (CoRL)*, 2019.

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

[10] Daniel S Brown, Yuchen Cui, and Scott Niekum. Risk-aware active inverse reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2018.

[11] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automaticaly-ranked demonstrations. In *Conference on Robot Learning (CoRL)*, 2019.

[12] Daniel S. Brown, Wonjoon Goo, Nagarajan Prabhat, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 2019.

[13] Daniel S. Brown and Scott Niekum. Efficient Probabilistic Performance Bounds for Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[14] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Advances in Neural Information Processing Systems*, 2015.

[15] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[16] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[17] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[18] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

[19] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, 2016.

[20] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[22] Josiah Hanna, Scott Niekum, and Peter Stone. Importance sampling policy evaluation with an estimated behavior policy. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, June 2019.

[23] Josiah P Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[24] Josiah P Hanna, Peter Stone, and Scott Niekum. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, 2017.

[25] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[26] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[27] Philippe Jorion. *Value at risk*. McGraw-Hill, New York, 1997.

[28] Jonathan Lacotte, Mohammad Ghavamzadeh, Yinlam Chow, and Marco Pavone. Risk-sensitive generative adversarial imitation learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2154–2163, 2019.

[29] M Laskey, J Lee, R Fox, A Dragan, and K Goldberg. Dart: Noise injection for robust imitation learning. *Conference on Robot Learning (CoRL)*, 2017.

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

[31] Alireza Makhzani and Brendan J Frey. Pixelgan autoencoders. In *Advances in Neural Information Processing Systems*, pages 1975–1985, 2017.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[33] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 663–670, 2000.

[34] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871, 2015.

[35] Marek Petrik and Reazul Hasan Russell. Beyond confidence regions: Tight bayesian ambiguity sets for robust mdps. *arXiv preprint arXiv:1902.07605*, 2019.

[36] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[37] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, pages 2586–2591, 2007.

[38] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

Algorithm 1 B-REX: Bayesian Reward Extrapolation

1: **input:** demonstrations D, pairwise preferences $\mathcal{P}$, MCMC proposal width σ, number of proposals to generate N, deep network architecture R_θ, and prior $P(w)$.
2: Pretrain R_θ using auxiliary tasks (see Section 5.2).
3: Freeze all but last layer, w, of R_θ. Let $\phi(s)$ be the activations of the penultimate layer of R_θ.
4: Precompute and cache $\Phi_\tau = \sum_{s \in \tau} \phi(s)$ for all $\tau \in D$.
5: Initialize w randomly.
6: Chain[0] $\leftarrow w$
7: Compute $P(\mathcal{P}, D|w)P(w)$ using Equation (7)
8: **for** $i \leftarrow 1$ to N **do**
9:     $\tilde{w} \leftarrow \mathcal{N}(w, \sigma)$ ▷ Normal proposal distribution
10:    Compute $P(\mathcal{P}, D|\tilde{w})P(\tilde{w})$ using Equation (7)
11:    $u \leftarrow \mathtt{Uniform}(0, 1)$
12:    **if** $u < \dfrac{P(\mathcal{P}, D|\tilde{w})P(\tilde{w})}{P(\mathcal{P}, D|w)P(w)}$ **then**
13:        Chain[i] $\leftarrow \tilde{w}$
14:        $w \leftarrow \tilde{w}$
15:    **else**
16:        Chain[i] $\leftarrow w$
17: **return** Chain

---

[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[40] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[41] Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the cvar via sampling. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2993–2999, 2015.

[42] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[43] Brijen Thananjeyan, Ashwin Balakrishna, Ugo Rosolia, Felix Li, Rowan McAllister, Joseph E Gonzalez, Sergey Levine, Francesco Borrelli, and Ken Goldberg. Extending deep model predictive control with safety augmented value estimation from demonstrations. *arXiv preprint arXiv:1905.13402*, 2019.

[44] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3000–3006, 2015.

[45] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, July 2018.

[46] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end simulated driving. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[47] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.

A   Preference-based Bayesian IRL

Pseudo-code for the optimized version of Bayesian Reward Extrapolation (B-REX) is shown in Algorithm 1.
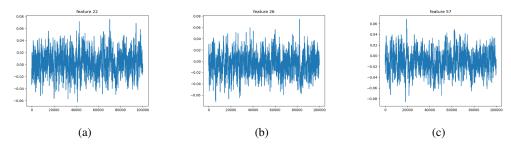
Figure 2: MCMC mixing plots for three randomly chosen weight features. Total dimensionality of weight vector is 65. MCMC was performed for 100,000 steps with a proposal width of $\sigma = 0.005$. Weights are normalized so that $\|w\|_1 = 1$.
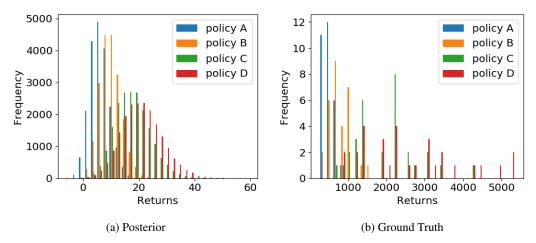

Figure 3: Beam Rider return distributions over the posterior $P(R|D, \mathcal{P})$ compared with the ground truth returns using game scores. Policies A-D correspond to checkpoints of an RL policy partially trained on the ground-truth reward function and correspond to 25, 325, 800, and 1450 training updates to PPO.

B   Mixing plots

The mixing plots for three randomly chosen features for Breakout are shown in Figure 2 and appear to be rapidly mixing.

C   Plots of return distributions for MCMC chain and ground-truth rewards

Figures 3–6 show the predicted and ground-truth distributions for different evaluation policies for Beam Rider, Enduro, Seaquest, and Space Invaders.

D   High-Confidence Lower Bounds (Full results)

We report the full results for computing high-confidence lower bounds on policy performance for Beam Rider, Enduro, Seaquest, and Space Invaders in tables 3–6.
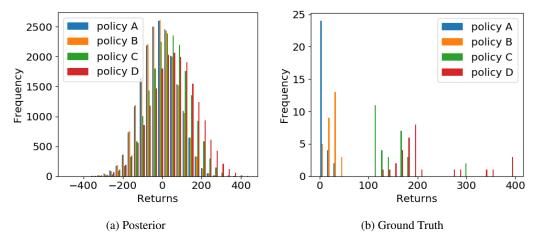
(a) Posterior   (b) Ground Truth

Figure 4: Enduro return distributions over the posterior $P(R|D, \mathcal{P})$ compared with the ground truth returns using game scores. Policies A-D correspond to checkpoints of an RL policy partially trained on the ground-truth reward function and correspond to 3125, 3425, 3900, 4875 training updates to PPO.
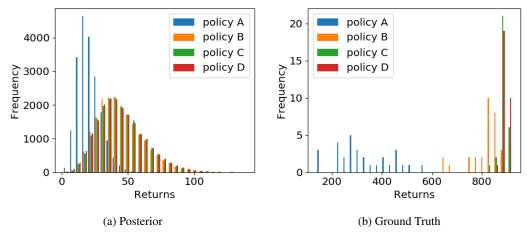

(a) Posterior   (b) Ground Truth

Figure 5: Seaquest return distributions over the posterior $P(R|D, \mathcal{P})$ compared with the ground truth returns using game scores. Policies A-D correspond to checkpoints of an RL policy partially trained on the ground-truth reward function and correspond to 25, 325, 800, and 1450 training updates to PPO.

Table 3: Policy Evaluation for Beam Rider.

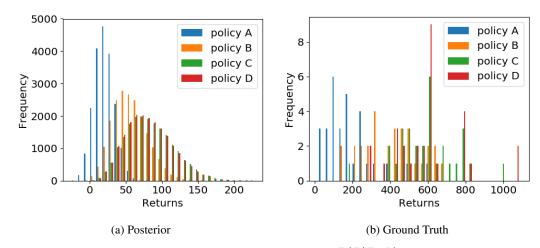| Policy	Mean Chain	0.05-VaR Chain	Traj. Length	GT Avg. Return	GT Min. Return
policy A	6.2	0.8	1398.8	471.3	264
policy B	9.9	4.2	1522.9	772.3	396
policy C	17.6	8.2	2527.5	1933.3	660
policy D	20.7	9.6	2963.2	2618.5	852
mean	97.4	61.2	8228.4	878.7	44
MAP	76.7	46.5	7271.0	1842.6	264

13

(a) Posterior   (b) Ground Truth

Figure 6: Space Invaders return distributions over the posterior $P(R|D, \mathcal{P})$ compared with the ground truth returns using game scores. Policies A-D correspond to checkpoints of an RL policy partially trained on the ground-truth reward function and correspond to 25, 325, 800, and 1450 training updates to PPO.

Table 4: Policy Evaluation for Enduro.

Policy	Mean Chain	0.05-VaR Chain	Traj. Length	GT Avg. Return	GT Min. Return
policy A	-12.0	-163.0	3322.1	7.7	0
policy B	-12.9	-163.9	3322.1	25.8	1
policy C	35.3	-127.9	3544.0	149.2	106
policy D	56.7	-132.8	4098.7	215.6	129
mean	164.1	-156.0	6761.1	450.1	389
MAP	208.6	-176.5	8092.3	569.7	441

Table 5: Policy Evaluation for Seaquest.

Policy	Mean Chain	0.05-VaR Chain	Traj. Length	GT Avg. Return	GT Min. Return
policy A	21.3	9.6	1055.9	327.3	140
policy B	43.9	19.8	2195.7	813.3	640
policy C	45.0	20.3	2261.8	880.7	820
policy D	44.6	19.3	2265.5	886.7	860
mean	63.1	27.5	2560.0	967.3	740
MAP	56.8	24.1	2249.1	570.7	520

Table 6: Policy Evaluation for Space Invaders.

Policy	Mean Chain	0.05-VaR Chain	Traj. Length	GT Avg. Return	GT Min. Return
policy A	21.5	-0.4	527.3	186.0	20
policy B	55.8	20.4	715.5	409.2	135
policy C	82.3	34.0	876.7	594.0	165
policy D	81.2	34.0	824.7	602.3	270
mean	257.2	113.6	2228.7	1437.5	515
MAP	246.2	109.0	2100.4	1440.2	600