Policy Gradient Bayesian Robust Optimization for Imitation Learning

Zaynah Javed * [1] Daniel S. Brown * [1] Satvik Sharma [1] Jerry Zhu [1] Ashwin Balakrishna [1] Marek Petrik [2]
Anca D. Dragan [1] Ken Goldberg [1]

Abstract

The difficulty in specifying rewards for many real-world problems has led to an increased focus on learning rewards from human feedback, such as demonstrations. However, there are often many different reward functions that explain the human feedback, leaving agents with *uncertainty* over what the true reward function is. While most policy optimization approaches handle this uncertainty by optimizing for expected performance, many applications demand risk-averse behavior. We derive a novel policy gradient-style robust optimization approach, PG-BROIL, that optimizes a soft-robust objective that balances expected performance and risk. To the best of our knowledge, PG-BROIL is the first policy optimization algorithm robust to a distribution of reward hypotheses which can scale to continuous MDPs. Results suggest that PG-BROIL can produce a family of behaviors ranging from risk-neutral to risk-averse and outperforms state-of-the-art imitation learning algorithms when learning from ambiguous demonstrations by hedging against uncertainty, rather than seeking to uniquely identify the demonstrator's reward function.

1. Introduction

We consider the following question: *How should an intelligent agent act if it has epistemic uncertainty over its objective function?* In the fields of reinforcement learning (RL) and optimal control, researchers and practitioners typically assume a known reward or cost function, which is then optimized to obtain a policy. However, even in settings where the reward function is specified, it is usually only a best approximation of the objective function that a human thinks will lead to desirable behavior. Furthermore, human-designed reward functions are also often augmented with human feedback. This may also result in reward uncertainty since human feedback, be it in the form of policy shaping (Griffith et al., 2013), reward shaping (Knox & Stone, 2012), or a hand-designed reward function (Hadfield-Menell et al., 2017; Ratner et al., 2018), can fail to perfectly disambiguate the human's intent true (Amodei et al., 2016).

Reward function ambiguity is also a key problem in imitation learning (Hussein et al., 2017; Osa et al., 2018), in which an agent seeks to learn a policy from demonstrations without access to the reward function that motivated the demonstrations. While many imitation learning approaches either sidestep learning a reward function and directly seek to imitate demonstrations (Pomerleau, 1991; Torabi et al., 2018) or take a maximum likelihood (Choi & Kim, 2011; Brown et al., 2019) or maximum entropy approach to learning a reward function (Ziebart et al., 2008; Fu et al., 2017), we believe that an imitation learning agent should explicitly reason about uncertainty over the true reward function to avoid misalignment with the demonstrator's objectives (Hadfield-Menell et al., 2017; Brown et al., 2020a). Bayesian inverse reinforcement learning (IRL) methods (Ramachandran & Amir, 2007) seek a posterior distribution over likely reward functions given demonstrations, but often perform policy optimization using the expected reward function or MAP reward function (Ramachandran & Amir, 2007; Choi & Kim, 2011; Ratner et al., 2018; Brown et al., 2020a). However, in many real world settings such as robotics, finance, and healthcare, we desire a policy which is robust to uncertainty over the true reward function.

Prior work on risk-averse and robust policy optimization in reinforcement learning has mainly focused on robustness to uncertainty over the true dynamics of the environment, but assumes a known reward function (Garcıa & Fernández, 2015; Tamar et al., 2015; Tang et al., 2020; Derman et al., 2018; Lobo et al., 2020; Thananjeyan et al., 2021). Some work addresses robust policy optimization under reward function uncertainty by taking a maxmin approach and optimizing a policy that is robust under the worst-case reward function (Syed et al., 2008; Regan & Boutilier, 2009; Hadfield-Menell et al., 2017; Huang et al., 2018). However, these approaches are limited to tabular domains, and maxmin approaches have been shown to sometimes lead to

[1] EECS Department, University of California, Berkeley [2] CS Department, University of New Hampshire. Correspondence to: Zaynah Javed <zjaved@berkeley.edu>, Daniel Brown <dsbrown@berkeley.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

incorrect and overly pessimistic policy evaluations (Brown & Niekum, 2018). As an alternative to maxmin approaches, recent work (Brown et al., 2020b) proposed a linear programming approach, BROIL: Bayesian Robust Optimization for Imitation Learning, that balances risk-aversion (in terms of Conditional Value at Risk (Rockafellar et al., 2000)) and expected performance. This approach supports a family of solutions depending on the risk-sensitivity of the application domain. However, as their approach is built on linear programming, it cannot be applied in MDPs with continuous state and action spaces and unknown dynamics.

In this work, we introduce a novel policy optimization approach that enables varying degrees of risk-sensitivity by reasoning about reward uncertainity while scaling to continuous MDPs with unknown dynamics. As in Brown et al. (2020b), we present an approach which reasons simultaneously about risk-aversion (in terms of Conditional Value at Risk (Rockafellar et al., 2000)) and expected performance and balances the two. However, to enable such reasoning in continuous spaces, we make a key observation: the Conditional Value at Risk objective supports efficient computation of an approximate subgradient, which can then be used in a policy gradient method. This makes it possible to use any policy gradient algorithm, such as TRPO (Schulman et al., 2017a) or PPO (Schulman et al., 2017b) to learn policies which are robust to reward uncertainity, resulting in an efficient and scalable algorithm. To the best of our knowledge, our proposed algorithm, Policy Gradient Bayesian Robust Optimization for Imitation Learning (PG-BROIL), is the first policy optimization algorithm robust to a distribution of reward hypotheses that can scale to complex MDPs with continuous state and action spaces.

To evaluate PG-BROIL, we consider settings where there is uncertainty over the true reward function. We first examine the setting where we have an a priori distribution over reward functions and find that PG-BROIL is able to optimize policies that effectively trade-off between expected and worst-case performance. Then, we leverage recent advances in efficient Bayesian reward inference (Brown et al., 2020a) to infer a posterior over reward functions from preferences over demonstrated trajectories. While other approaches which do not reason about reward uncertainty overfit to a single reward function hypothesis, PG-BROIL optimizes a policy that hedges against multiple reward function hypotheses. When there is high reward function ambiguity due to limited demonstrations, we find that PG-BROIL results in significant performance improvements over other state-of-the-art imitation learning methods.

2. Related Work

Reinforcement Learning: There has been significant recent interest in safe and robust reinforcement learning (Garcıa & Fernández, 2015); however, most approaches are only robust with respect to noise in transition dynamics and only consider optimizing a policy with respect to a single reward function. Existing approaches reason about risk measures with respect to a single task rewards (Heger, 1994; Shen et al., 2014; Tamar et al., 2014; Tang et al., 2019), establish convergence to safe regions of the MDP (Thananjeyan et al., 2020b;a), or optimize a policy to avoid constraint violations (Achiam et al., 2017; Fisac et al., 2018; Thananjeyan et al., 2021).

In this paper, we develop a reinforcement learning algorithm which reasons about risk with respect to a belief distribution over the task reward function. We focus on being robust to tail risk by optimizing for conditional value at risk (Rockafellar et al., 2000). However, unlike prior work (Heger, 1994; Shen et al., 2014; Tamar et al., 2014; 2015; Tang et al., 2019; Zhang et al., 2021), which focuses on risk with respect to a known reward function and stochastic transitions, we consider policy optimization when there is epistemic uncertainty over the reward function itself. We formulate a soft-robustness approach that blends optimizing for expected performance and optimizing for the conditional value at risk. Recent work also considers soft-robust objectives when there is uncertainty over the correct transition model of the MDP (Lobo et al., 2020; Russel et al., 2020), rather than uncertainty over the true reward function.

Imitation Learning: Imitation learning approaches vary widely in reasoning about reward uncertainty. Behavioral cloning approaches simply learn to imitate the actions of the demonstrator, resulting in quadratic regret (Ross & Bagnell, 2010). DAgger (Ross et al., 2011) achieves sublinear regret by repeatedly soliciting human action labels in an online fashion. While there has been work on safe variants of DAgger (Zhang & Cho, 2016; Hoque et al., 2021), these methods only enable robust policy learning by asymptotically converging to the policy of the demonstrator, and always assume access to an expert human supervisor.

Inverse reinforcement learning (IRL) methods are another way of performing imitation learning (Arora & Doshi, 2018), where the learning agent seeks to achieve better sample efficiency and generalization by learning a reward function which is then optimized to obtain a policy. However, most inverse reinforcement learning methods only result in a point-estimate of the demonstrator's reward function (Abbeel & Ng, 2004; Ziebart et al., 2008; Fu et al., 2017; Brown et al., 2019). Risk-sensitive IRL methods (Lacotte et al., 2018; Majumdar et al., 2017; Santara et al., 2018) assume risk-averse experts and focus on optimizing policies that match the risk-aversion of the demonstrator; however, these methods focus on the aleatoric risk induced by transition probabilities and there is no clear way to adapt risk-averse IRL to the Bayesian robust setting, where the objective is to be robust to epistemic risk over reward hypotheses rather than risk with respect to stochasticity in the dynamics. Bayesian IRL approaches explicitly learn a distribution over reward functions conditioned on the demonstrations, but usually only optimize a policy for the expected reward function or MAP reward function under this distribution (Ramachandran & Amir, 2007; Choi & Kim, 2011; Brown et al., 2020a).

We seek to optimize a policy that is robust to epistemic uncertainty in the true reward function of an MDP. Prior work on robust imitation learning has primarily focused on maxmin approaches which seek to optimize a policy for an adversarial worst-case reward function (Syed et al., 2008; Ho & Ermon, 2016; Regan & Boutilier, 2009; Hadfield-Menell et al., 2017; Huang et al., 2018). However, these approaches can learn overly pessimistic behaviors (Brown & Niekum, 2018) and existing approaches assume discrete MDPs with known transition dynamics (Syed et al., 2008; Regan & Boutilier, 2009; Hadfield-Menell et al., 2017) or require fully solving an MDP hundreds of times (Huang et al., 2018), effectively limiting these approaches to discrete domains. Recently, (Brown et al., 2020b) proposed a method for robust Bayesian optimization for imitation learning (BROIL), which optimizes a soft-robust objective that balances expected performance with conditional value at risk (Rockafellar et al., 2000). However, their approach is limited to discrete state and action spaces and known transition dynamics. By contrast, we derive a novel policy gradient approach which enables robust policy optimization with respect to reward function uncertainty for domains with continuous states and action and unknown dynamics.

3. Preliminaries and Notation

3.1. Markov Decision Processes

We model the environment as a Markov Decision Process (MDP) (Puterman, 2005). An MDP is a tuple $(\mathcal{S}, \mathcal{A}, r, P, \gamma, p_0)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, transition dynamics $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, discount factor $\gamma \in [0, 1)$, and initial state distribution p_0. We consider stochastic policies $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ which output a distribution over $\mathcal{A}$ conditioned on a state $s \in \mathcal{S}$. We denote the expected return of a policy π under reward function r as $v(\pi, r) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$.

3.2. Distributions over Reward Functions

We are interested in solving MDPs when there is epistemic uncertainty over the true reward function. When we refer to the reward function as a random variable we will use R, and will use r to denote a specific model of the reward function. Reward functions are often parameterized as a linear combination of known features (Abbeel & Ng, 2004; Ziebart et al., 2008; Sadigh et al., 2017) or as a deep neural network (Ho & Ermon, 2016; Fu et al., 2017). Thus, we can model uncertainty in the reward function as a distribution over R, or, equivalently, as a distribution over the reward function parameters. This distribution could be a prior distribution $\mathbb{P}(R)$ that the agent learns from previous tasks (Xu et al., 2019). Alternatively, the distribution could be the posterior distribution $\mathbb{P}(R \mid D)$ learned via Bayesian inverse reinforcement learning (Ramachandran & Amir, 2007) given demonstrations D, the posterior distribution $\mathbb{P}(R \mid \mathcal{P}, D)$ given preferences $\mathcal{P}$ over demonstrations (Sadigh et al., 2017; Brown et al., 2020a), or the posterior distribution $\mathbb{P}(R \mid r')$ learned via inverse reward design given a human-specified proxy reward r' (Hadfield-Menell et al., 2017; Ratner et al., 2018). This distribution is typically only available via sampling techniques such as Markov chain Monte Carlo (MCMC) sampling (Ramachandran & Amir, 2007; Hadfield-Menell et al., 2017; Brown et al., 2020a).

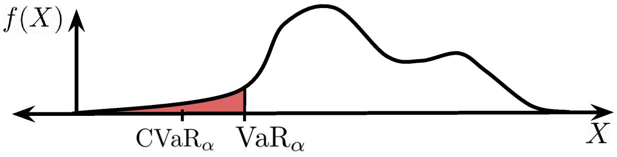


Figure 1. The pdf $f(X)$ of a random variable X. VaR_α measures the $(1-\alpha)$-quantile outcome. CVaR_α measures the expectation given that we only consider values less than the VaR_α.

3.3. Risk Measures

We are interested in robust policy optimization with respect to a distribution over the performance of the policy induced by a distribution over possible reward functions. Consider a policy π and a reward distribution $\mathbb{P}(R)$. Together, π and $\mathbb{P}(R)$ induce a distribution over the expected return of the policy, $v(\pi, R), R \sim \mathbb{P}(R)$. We seek a robust policy that minimizes tail risk, given some risk measure, under the induced distribution v. Figure 1 visualizes two common risk measures: value at risk (VaR) and conditional value at risk (CVaR), for a general random variable X. In our setting, X corresponds to the expected return, $v(\pi, R)$, of a policy π under the reward function random variable R, and the objective is to minimize the tail risk (visualized in red).

3.3.1. Value at Risk

Given a risk-aversion parameter $\alpha \in [0, 1]$, the VaR_α of a random variable X is the $(1-\alpha)$-quantile outcome:

$$\mathrm{VaR}_\alpha[X] = \sup\{x : \mathbb{P}(X \geq x) \geq \alpha\}, \tag{1}$$

where it is common to have $\alpha \in [0.9, 1]$.

Despite the popularity of VaR, optimizing a policy for VaR has several problems: (1) optimizing for VaR results in an NP hard optimization problem (Delage & Mannor, 2010),

(2) VaR ignores risk in the tail that occurs with probability less than $(1-\alpha)$ which is problematic for domains where there are rare but potentially catastrophic outcomes, and (3) VaR is not a coherent risk measure (Artzner et al., 1999).

3.3.2. Conditional Value at Risk

CVaR is a coherent risk measure (Delbaen, 2002), also known as average value at risk, expected tail risk, or expected shortfall. For continuous distributions

$$\mathrm{CVaR}_\alpha[X] = \mathbb{E}_{f(X)}\left[X \mid X \leq \mathrm{VaR}_\alpha[X]\right]. \quad (2)$$

In addition to being coherent, CVaR can be maximized via convex optimization, does not ignore the tail of the distribution, and is a lower bound on VaR. Because of these desirable properties, we would like to use CVaR as our risk measure. However, because posterior distributions obtained via Bayesian IRL are often discrete (Ramachandran & Amir, 2007; Sadigh et al., 2017; Hadfield-Menell et al., 2017; Brown & Niekum, 2018), we cannot directly optimize for CVaR using the definition in Equation (2) since this definition only works for atomless distributions. Instead, we make use of the following definition of CVaR, proposed by Rockafellar et al. (2000), that works for any distribution:

$$\mathrm{CVaR}_\alpha[X] = \max_\sigma \left(\sigma - \frac{1}{1-\alpha}\mathbb{E}[(\sigma - X)_+]\right), \quad (3)$$

where $(x)_+ = \max(0, x)$ and σ roughly corresponds to the VaR_α. To gain intuition for this formula, note that if we define $\sigma = \mathrm{VaR}_\alpha[X]$ we can rewrite CVaR_α as

$$\begin{aligned} \mathrm{CVaR}_\alpha[X] &= \mathbb{E}_{f(X)}[X \mid X \leq \sigma] & (4)\\ &= \sigma - \mathbb{E}_{f(X)}[\sigma - X \mid X \leq \sigma] & (5)\\ &= \sigma - \frac{\mathbb{E}_{f(X)}[\mathbf{1}_{X\leq\sigma}\cdot(\sigma - X)]}{P(X \leq \sigma)} & (6)\\ &= \sigma - \frac{1}{1-\alpha}\mathbb{E}_{f(X)}[(\sigma - X)_+] & (7)\end{aligned}$$

where $\mathbf{1}_x = 1$ is the indicator function that evaluates to 1 if x is True and 0 otherwise, and where we used the linearity of expectation, the definition of conditional expectation, and the definitions of $\mathrm{VaR}_\alpha[X]$, and $(x)_+$. Taking the maximum over $\sigma \in \mathbb{R}$, gives us the definition in Equation (3).

4. Bayesian Robust Optimization for Imitation Learning

In Section 4.1 we describe the Bayesian robust optimization for imitation learning (BROIL) objective, previously proposed by (Brown et al., 2020b). Then, in sections 4.2 and 4.3, we derive a novel policy gradient update for BROIL and provide an intuitive explanation for the result.

4.1. Soft-Robust BROIL Objective

Rather than seeking a purely risk-sensitive or purely risk-neutral approach, we seek to optimize a soft-robust objective that balances the expected and probabilistic worst-case performance of a policy. Given some performance metric $\psi(\pi_\theta, R)$ where $R \sim \mathbb{P}(R)$, Brown et al. (2020b) recently proposed Bayesian Robust Optimization for Imitation Learning (BROIL) which seeks to optimize the following:

$$\max_{\pi_\theta} \lambda\cdot\mathbb{E}_{\mathbb{P}(R)}[\psi(\pi_\theta, R)] + (1-\lambda)\cdot\mathrm{CVaR}_\alpha\left[\psi(\pi_\theta, R)\right] \quad (8)$$

For MDPs with discrete states and actions and known dynamics, Brown et al. (2020b) showed that this problem can be formulated as a linear program which can be solved in polynomial time. However, many MDPs of interest involve continuous states and actions and unknown dynamics.

4.2. BROIL Policy Gradient

We now derive a policy gradient objective for BROIL that allows us to extend BROIL to continuous states and actions and unknown transition dynamics, enabling robust policy learning in a wide variety of practical settings. Given a parameterized policy π_θ and N possible reward hypotheses, there are many possible choices for the performance metric $\psi(\pi_\theta, R)$. Brown et al. (2020a) considered two common metrics: (1) expected value, i.e., $\psi(\pi_\theta, R) = v(\pi, R) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$ and (2) baseline regret, i.e., $\psi(\pi_\theta, R) = v(\pi_\theta, R) - v(\pi_E, R)$ where π_E denotes an expert policy (usually estimated from demonstrations). In Appendix A we derive a more general form for any performance metric $\psi(\pi_\theta, R)$ and also give the derivation for the baseline regret performance metric. For simplicity, we let $\psi(\pi_\theta, R) = v(\pi, R)$ (expected return) hereafter.

To find the policy that maximizes Equation (8) we need the gradient with respect to the policy parameters θ. For the first term in Equation (8), we have

$$\nabla_\theta \mathbb{E}_{\mathbb{P}(R)}[v(\pi_\theta, R)] \approx \sum_{i=1}^N \mathbb{P}(r_i)\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)]. \quad (9)$$

Next, we consider the gradient of the CVaR term. CVaR is not differentiable everywhere so we derive a sub-gradient. Given a finite number of samples from the reward function posterior, we can write this sub-gradient as

$$\nabla_\theta \max_\sigma \Big(\sigma - \frac{1}{1-\alpha}\sum_{i=1}^N \mathbb{P}(r_i)\big(\sigma - \mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)]\big)_+\Big) \quad (10)$$

where $(x)_+ = \max(0, x)$. To solve for the sub-gradient of this term, note that given a fixed policy π_θ, we can solve for σ via a line search: since the objective is piece-wise linear we only need to check the value at each point $v(\pi, r_i)$, for each reward function sample from the posterior since these are the endpoints of each linear segment. If we let $v_i = v(\pi, r_i)$ then we can quickly iterate over all reward function hypotheses and solve for σ as

$$\sigma^* = \underset{\sigma \in \{v_1, \dots, v_N\}}{\operatorname{argmax}} \Big(\sigma - \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbb{P}(r_i) \big[\sigma - v_i\big]_+\Big). \quad (11)$$

Solving for σ^* requires estimating v_i by collecting a set $\mathcal{T}$ of on-policy trajectories $\tau \sim \pi_\theta$ where $\tau = (s_0, a_0, s_1, a_1, \dots, s_T, a_T)$:

$$v_i \approx \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} r_i(s_t, a_t). \quad (12)$$

Solving for σ^* does not require additional data collection beyond what is required for standard policy gradient approaches. We simply evaluate the set of rollouts $\mathcal{T}$ from π_θ under each reward function hypothesis, r_i and then solve the optimization problem above to find σ^*. While this requires more computation than a standard policy gradient approach—we have to evaluate each rollout under N reward functions—this does not increase the online data collection, which is often the bottleneck in RL algorithms.

Given the solution σ^* found by solving the optimization problem in (11), we perform a step of policy gradient optimization by following the sub-gradient of CVaR with respect to the policy parameters θ:

$$\nabla_\theta \, \text{CVaR}_\alpha = \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbb{P}(r_i) \mathbf{1}_{\sigma^* \geq v(\pi_\theta, r_i)} \nabla_\theta v(\pi_\theta, r_i) \quad (13)$$

where $\mathbf{1}_x$ is the indicator function that evaluates to 1 if x is True and 0 otherwise. Given the sub-gradient of the BROIL objective (13), the only thing remaining to compute is the standard policy gradient. Note that in standard RL, we write the policy gradient as (Sutton & Barto, 2018):

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t(\tau) \right] \quad (14)$$

where Φ_t is a measure of the performance of trajectory τ starting at time t. One of the most common forms of $\Phi_t(\tau)$ is the on-policy advantage function (Schulman et al., 2015) with respect to some single reward function:

$$\Phi_t(\tau) = A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t). \quad (15)$$

If we define $\Phi_t^{r_i}$ in terms of a particular reward function r_i, then, as we show in Appendix A, we can rearrange terms in the standard policy gradient formula to obtain the following form for the BROIL policy gradient which we estimate using a set $\mathcal{T}$ of on-policy trajectories $\tau \sim \pi_\theta$ where $\tau = (s_0, a_0, s_1, a_1, \dots, s_T, a_T)$ as follows:

$$\nabla_\theta \text{BROIL} \approx \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) w_t(\tau) \right] \quad (16)$$

where

$$w_t(\tau) = \sum_{i=1}^{N} \mathbb{P}(r_i) \Phi_t^{r_i}(\tau) \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq v(\pi, r_i)} \right) \quad (17)$$

is the weight associated with each state-action pair (s_t, a_t) in the set of trajectory rollouts $\mathcal{T}$. The resulting vanilla policy gradient algorithm is summarized in Algorithm 1. In Appendix C we show how to apply a trust-region update based on Proximal Policy Optimization (Schulman et al., 2017b) for more stable policy gradient optimization.

4.3. Intuitive Interpretation of the Policy Gradient

Consider the policy gradient weight w_t given in Equation (17). If $\lambda = 1$, then

$$w_t(\tau) = \sum_{i=1}^{N} \mathbb{P}(R_i) \Phi_t^{R_i}(\tau) = \Phi_t^{\bar{R}}(\tau) \quad (18)$$

where $\bar{R}$ is the expected reward under the posterior. Thus, $\lambda = 1$ is equivalent to standard policy gradient optimization under the mean reward function and gradient ascent will focus on increasing the likelihood of actions that look good in expectation over the reward function distribution $\mathbb{P}(R)$. Alternatively, if $\lambda = 0$, then

$$w_t(\tau) = \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbf{1}_{\sigma^* \geq v(\pi, R_i)} \mathbb{P}(R_i) \Phi_t^{R_i}(\tau) \quad (19)$$

and gradient ascent will increase the likelihood of actions that look good under reward functions that the current policy π_θ performs poorly under, i.e., policy gradient updates will focus on improving performance under all R_i such that $v(\pi, R_i) \leq \sigma^*$, weighting the gradient according to the likelihood of these worst-case reward functions. The update rule also multiplies by $1/(1-\alpha)$ which acts to normalize the magnitude of the gradient: as $\alpha \to 1$ we update on reward functions further into the tail, which have smaller probability mass. Thus, $\lambda \in [0, 1]$ allows us to blend between maximizing policy performance in expectation versus worst-case and $\alpha \in [0, 1)$ determines how far into the tail of the distribution to focus the worst-case updates.

Algorithm 1 Policy Gradient BROIL

```
Input: initial policy parameters $\theta_0$, samples from reward function posterior $r_1, \dots, r_N$ and associated probabilities, $\mathbb{P}(r_1), \dots, \mathbb{P}(r_N)$.
for $k = 0, 1, 2, \dots$ do
    Collect set of trajectories $\mathcal{T}_k = \{\tau_i\}$ by running policy $\pi_{\theta_k}$ in the environment.
    Estimate expected return of $\pi_{\theta_k}$ under each reward function hypothesis $r_j$ using Eq. (12).
    Solve for $\sigma^*$ using Eq. (11)
    Estimate policy gradient using Eq. (16) and Eq. (17).
    Update $\theta$ using gradient ascent.
end for
```

5. Experiments

In experiments, we consider the following questions: (1) Can PG-BROIL learn control policies in MDPs with continuous states and actions and unknown transition dynamics? (2) Does optimizing PG-BROIL with different values of λ effectively trade-off between maximizing for expected return and maximizing robustness? (3) When demonstrations are ambiguous, can PG-BROIL outperform other imitation learning baselines by hedging against uncertainty?

Code and videos are available at `https://sites.google.com/view/pg-broil`.

5.1. Prior over Reward Functions

We first consider an RL agent with a priori uncertainty over the true reward function. This setting allows us to initially avoid the difficulties of inferring a posterior distribution over reward functions and carefully examine whether PG-BROIL can trade-off expected performance and robustness (CVaR) under epistemic uncertainty over the true reward function. We study 3 domains: the classical CartPole benchmark (Brockman et al., 2016), a pointmass navigation task inspired by (Thananjeyan et al., 2020b) and a robotic reaching task from the from the DM Control Suite (Tassa et al., 2020). All domains are characterized by a robot navigating in an environment where some states have uncertain costs. All domains have unknown transition dynamics and continuous states and actions (except CartPole which has discrete actions). We implement PG- BROIL on top of OpenAI Spinning Up (Achiam, 2018). For cartpole we implement PG-BROIL on top of REINFORCE (Peters & Schaal, 2008) and for remaining domains we implement PG-BROIL on top of PPO (Schulman et al., 2017b) (see Appendix C).

5.1.1. Experimental Domains

CartPole: We consider a risk-sensitive version of the classic CartPole benchmark (Brockman et al., 2016). The reward function is $R(s) = b \cdot s_x$, where s_x is the position of the cart on the track, and there is uncertainty over b. Our prior over b is distributed uniformly in the range [-1, 0.2]. The center of the track is $s_x = 0$. We sample values of b between -1 and 0.2 across even intervals of 0.2 width to form a discrete posterior distribution for PG-BROIL. The reward distribution is visualized in Figure 2a. Based on our prior distribution over reward functions, the left side of the track ($s_x < 0$) is associated with a higher expected reward but a worse worst case scenario (the potential for negative rewards). By contrast, the robust solution is to stay in the middle of the track in order to perform well across all possible reward functions since the center of the track has less risk of a significantly negative reward than the left or right sides of the track.

Pointmass Navigation: We next consider a risk-sensitive continuous 2-D navigation task inspired by Thananjeyan et al. (2020b). Here the objective is to control a pointmass robot towards a known goal location with forces in cardinal directions in a system with linear Gaussian dynamics and drag. There are gray regions of uncertain cost that can either be traversed or avoided as illustrated in Figure 2b. For example, these regions could represent grassy areas which are likely easy to navigate, but where the grass may occlude mud or holes which would impede progress and potentially cause damage or undue wear and tear on the robot. The robot has prior knowledge that it needs to reach the goal location $g = (0, 0)$ on the map, depicted by the red star. We represent this prior with a nominal cost for each step that is the distance to the goal from the robot's position. We add a penalty term of uncertain cost for going through the gray region giving the following reward function posterior:

$$R(s) = -\left(\|s_{x,y} - g\|_2^2 + b \cdot \mathbf{1}_{\text{gray}}\right), b \sim \mathbb{P}(b), \tag{20}$$

where $\mathbf{1}_{\text{gray}}$ is an indicator for entering a gray region, and where the distribution $\mathbb{P}(b)$ over the penalty b is given as

b	-500	-40	0	40	50
$\mathbb{P}(b)$	0.05	0.05	0.2	0.3	0.4

On average it is favorable to go through the gray region ($\mathbb{E}[b] = +5$), but there is some probability that going through the gray region is highly unfavorable:

Reacher: We design a modified version of the Reacher environment from the DeepMind Control Suite (Tassa et al., 2020) (Figure 2c), which is a 2 link planar arm where the robot can apply joint torques to each of the 2 joints to guide the end effector of the arm to a goal position on the plane. We modify the original environment by including an area of uncertainty (large red circle). When outside the uncertain region, the robot receives a reward which penalizes the distance between the end effector and the goal (small yellow circle). Thus, the robot is normally incentivized to guide the end effector to the goal as quickly as possible. When the end

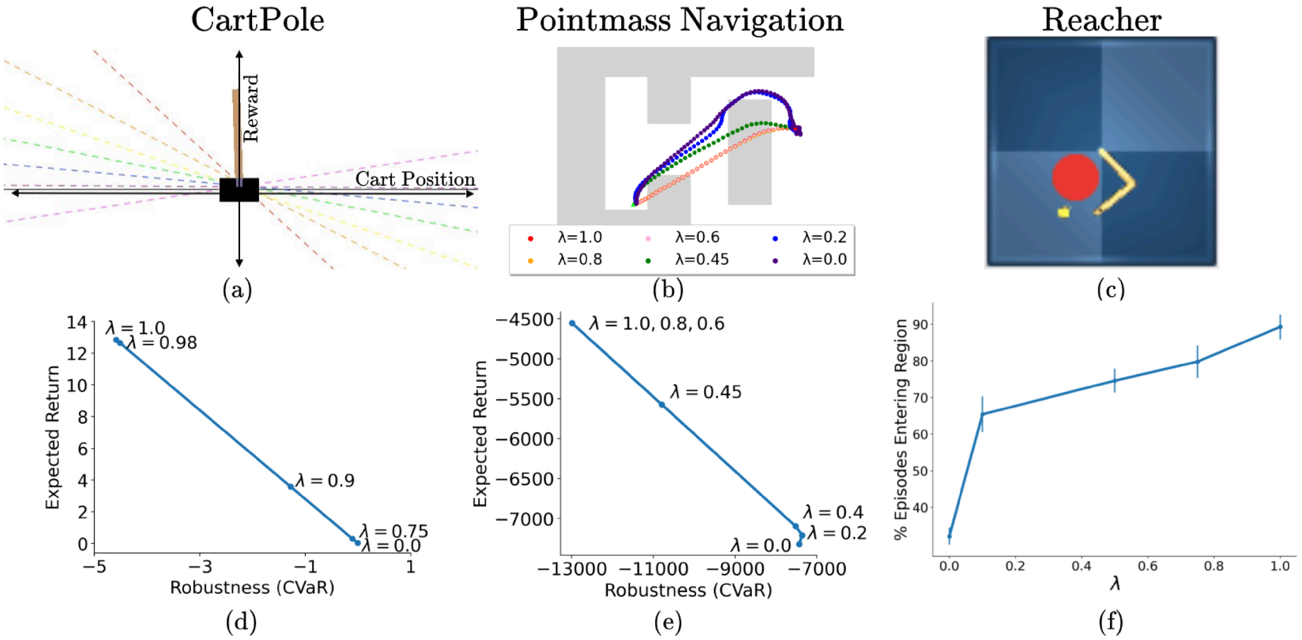


Figure 2. **Prior over Reward Functions: Domains and Results.** We study (a) CartPole in which the reward is an unknown linear function of the cart's position, (b) Pointmass Navigation with gray regions of uncertain costs, and (c) Reacher with a red region of uncertain cost. For the CartPole and Pointmass Navigation domains, we find that as λ is decreased, the learned policy optimizes more for being robust to tail risk and thus achieves more robust performance (in terms of CVaR) at the expense of expected return in panels (d) and (e). In panel (f), we find that the reacher arm enters the riskier red region less often with decreasing λ as expected.

effector is inside the uncertain region, the robot has an 80% chance of receiving a +2 bonus, a 10% chance of receiving a -2 penalty, and a 10% chance of neither happening (receiving rewards as if it were outside the uncertain region). The large red circle can be interpreted as a region on the table that has a small chance of causing harm to the robot or breaking an object on the table. However, in expectation the robot believes it is good to enter the red region (e.g., assuming that objects in this region are not fragile).

5.1.2. Results

PG-BROIL consistently exhibits more risk-averse behaviors with decreasing λ across all domains. For CartPole and Pointmass Navigation, we see that as λ is decreased, the learned policy becomes more robust to tail risk at the expense of lower expected return in Figures 2d and 2e respectively. Figure 2e indicates that values of λ close to 0 can lead to unstable policy optimization due to excessive focus on tail risk—the policy for $\lambda = 0$ is Pareto dominated by the policy for $\lambda = 0.2$. We visualize the learned behaviors for different values of λ for the Pointmass Navigation environment in Figure 2b. For high values of λ, the robot cuts straight through the uncertain terrain, for intermediate values (eg. $\lambda = 0.45$), the robot somewhat avoids the uncertain terrain, while for low values of λ, the robot almost entirely avoids the uncertain terrain at the expense of a longer path. Finally, for the Reacher environment, we find that the percentage of episodes where the arm enters the red region decreases as λ decreases as expected (Figure 2f).

5.2. Learning from Demonstrations

Our previous results demonstrated that PG-BROIL is able to learn policies that effectively balance expected performance and robustness in continuous MDPs under a given prior over reward functions. In this section, we consider the imitation learning setting where a robot infers a reward function from demonstrated examples. Given such input, there are typically many reward functions that are consistent with it; however, many reward inference algorithms (Fu et al., 2017; Finn et al., 2016; Brown et al., 2019) will output only one of them—not necessarily the true reward. There has been some work on Bayesian algorithms such as Bayesian IRL (Ramachandran & Amir, 2007) which estimates a *posterior distribution* instead of a single reward and Bayesian REX (Brown et al., 2020a) which makes it possible to efficiently learn this posterior from preferences over high dimensional demonstrated examples of varying qualities. However, prior work on Bayesian reward learning often only optimizes policies for the expected or MAP reward estimate over the learned posterior (Ramachandran & Amir, 2007; Choi & Kim, 2011; Brown et al., 2020a). Our hypothesis is that for imitation learning problems with high uncertainty about the true reward function, taking a robust optimization approach via PG-BROIL will lead to better

Table 1. **TrashBot:** We evaluate PG-BROIL against 5 other imitation learning algorithms when learning from ambiguous preferences over demonstrations (Figure 3). Results are averages (± one st. dev.) over 10 random seeds and 100 test episodes each with a horizon of 100 steps per episode. For PG-BROIL, we set $\alpha = 0.95$ and report results for the best λ ($\lambda = 0.8$).

ALGORITHM	AVG. TRASH COLLECTED	AVG. STEPS IN GRAY REGION
BC	3.4 ± 1.8	2.7 ± 6.2
GAIL	2.2 ± 1.5	3.7 ± 9.9
RAIL	1.1 ± 1.2	2.2 ± 6.9
PBRL	2.6 ± 1.5	1.2 ± 2.7
BAYESIAN REX	1.6 ± 1.3	1.2 ± 1.7
PG-BROIL	$\mathbf{8.4 \pm 0.5}$	$\mathbf{0.1 \pm 0.1}$

performance by producing policies that do well in expectation, but also avoid low reward under *any* of the sufficiently probable reward functions in the learned posterior.

5.2.1. TrashBot from Demos

We first consider a continuous control TrashBot domain (Figure 3), where aim to teach a robot to pick up pieces of trash (black dots) while avoiding the gray boundary regions. The state-space, dynamics and actions are the same as for the Pointmass Navigation environment and we provide human demonstrations via a simple teleoperation interface. The robot constructs its reward function hypotheses as linear combinations of three binary features which correspond to: (1) being in the gray region (GRAY), (2) being in the white region (WHITE), and (3) picking up a piece of trash (TRASH). We give three pairwise preferences over human teleoperated trajectories (generated by one of the authors) as shown in Figure 3. However, the small number of preferences makes it challenging for the robot to ascertain the true reward function parameters as there are many reward function weights that would lead to the same human preferences. Furthermore, the most salient feature is WHITE and this feature is highly correlated, but not causal, with the preferences. Thus, this domain can easily lead to reward hacking/gaming behaviors (Krakovna et al., 2020). We hypothesize that PG-BROIL will hedge against uncertainty and learn to pick up trash while avoiding the gray region.

We compare against behavioral cloning (BC), GAIL (Ho & Ermon, 2016), and Risk-Averse Imitation Learning (RAIL) (Santara et al., 2018), which estimates CVaR over trajectories to create a risk-averse version of the GAIL algorithm. To facilitate a fairer comparison, we only give BC, GAIL, and RAIL the better ranked demonstration from each preference pair. We also compare with Preference-based RL (PBRL) (Christiano et al., 2017) in the offline demonstration setting (Brown et al., 2019) which optimizes an MLE estimate of the reward weights and Bayesian REX (Brown et al., 2020a), which optimizes the mean reward function under the posterior distribution given the preferences. PG-BROIL also uses Bayesian REX (Brown et al., 2020a) to infer a reward function posterior distribution given the preferences over demonstrations (see Appendix E for details), but optimizes the BROIL objective.

Table 1 compares the performance of each baseline imitation learning algorithm when given the 3 pairs of demonstrations shown in Figure 3. We find that PG-BROIL outperforms BC and GAIL (Ho & Ermon, 2016) by not directly seeking to imitate the states and actions in the demonstrations, but by explicitly reasoning about uncertainty in the true reward function. We also find that PG-BROIL significantly outperforms RAIL. This is because RAIL only focuses on minimizing aleatoric uncertainty under stochastic transition dynamics for a single reward function (the discriminator), not epistemic uncertainty over the true reward function. We find that PG-BROIL outperforms PBRL and Bayesian REX.

We inspected the learned reward functions and found that the PBRL reward places heavy emphasis on collecting trash but has a small positive weight on the WHITE feature. We hypothesize that this results in policy optimization falling into a local maxima in which it mostly mines rewards by staying in the white region. By contrast, PG-BROIL considers a number of reward hypotheses, many of which have negative weights on the WHITE feature. Thus, a risk-averse agent cannot mine rewards by simply staying in the white region, and is incentivized to maximally pick up trash while keeping visits to the gray region low. The mean reward function optimized by Bayesian REX penalizes visiting the gray region but learns roughly equal weights for the WHITE and TRASH features. Thus, Bayesian REX is not strongly incentivized to pick up trash. Because of this the learned policy sometimes visits the borders of the white region and occasionally enters the gray region when it accumulates too high of a velocity. By contrast, PG-BROIL effectively optimizes a policy that is robust to multiple hypotheses that explain the rankings: picking up trash more than any other policy, while avoiding the gray region. See Appendix F.

5.2.2. Reacher from Demos with Domain Shift

For this experiment, we use the same Reacher environment described above. We give the agent five pairwise preferences over demonstrations of varying quality in a training domain where the uncertain reward region is never close to the goal and where none of the demonstrations show the reacher arm entering the uncertain region. We then introduce domain shift by both optimizing and testing policies in reacher environments unseen in the demonstrations, where the goal location is randomized and sometimes the uncertain reward region is in between the the reacher arm and the goal. The inferred reward function is a linear combination of 2

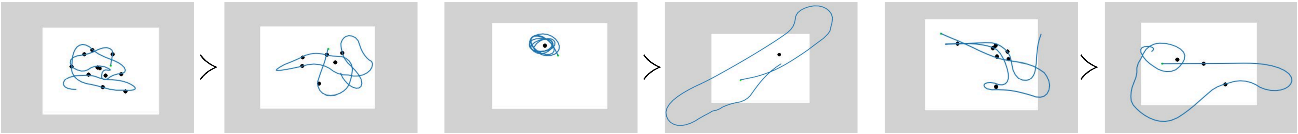


Figure 3. **TrashBot environment:** Each time the robot picks up a piece of trash (by moving close to a black dot), a new one appears at a randomly in the white region. We give pairwise preferences over human demos that aim to teach the robot that picking up trash is good (left), going into the gray region is undesirable (center), and less time in the gray region and picking up more trash is preferred (right).

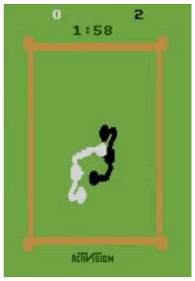
Table 2. **Reacher from Demos:** We evaluate PG-BROIL and baseline imitation learning algorithms when learning from preferences over demonstrations. Results are averages (± one st. dev.) over 3 seeds and 100 test episodes with a horizon of 200 steps per episode. For PG-BROIL, we set $\alpha = 0.9$ and report results for $\lambda = 0.15$.

ALGORITHM	AVG. STEPS IN UNCERTAIN REGION	AVG. STEPS IN TARGET REGION
BC	11.3 ± 27.4	39.9 ± 62.3
GAIL	2.3 ± 1.7	5.1 ± 13.0
RAIL	2.1 ± 1.2	4.6 ± 27.0
PBRL	28.4 ± 37.7	16.8 ± 30.4
BAYESIAN REX	13.5 ± 35.0	94.5 ± 70.1
PG-BROIL	**1.7 ± 7.2**	**102.0 ± 60.5**

features: TARGET and UNCERTAIN REGION which are simply binary indicators which identify whether the agent is in the target location or in the uncertain region respectively. In the posterior generated using Bayesian REX, we find that the weight learned for the TARGET feature is strongly positive over all reward functions. UNCERTAIN REGION, having no information from any of the demonstrations, has a wide variety of possible values from -1 to +1 (reward weights are normalized to have unit L2-norm). Both the mean and MLE reward functions assign a positive weight to both the TARGET and UNCERTAIN REGION features, resulting in Bayesian REX and PBRL frequently entering the uncertain region as shown in Table 2. By contrast, PG-BROIL hedges against its uncertainty over the quality of the uncertain region and avoids it. See Appendix D.3.

5.2.3. ATARI BOXING FROM DEMOS

For this experiment, we give the agent 3 preferences over suboptimal demos of the Atari Boxing game (Bellemare et al., 2013). We use Bayesian REX to infer a reward function posterior where each inferred reward functions is a linear combinations of 3 binary indicator features identifying whether the agent hit its opponent, got hit, or stayed away from the opponent. The mean and MLE reward functions both assign a high weight to hitting the opponent, ignoring the risk of getting hit by the opponent due to always staying close to the opponent in order to score hits on it. PG-BROIL tries to satisfy multiple reward functions by both trying to avoid getting hit and scoring hits, resulting in better performance under the true reward as shown in Table 4. See Appendix D.5 for more details.



(a)

ALGORITHM	GAME SCORE
BC	1.7 ± 5.3
GAIL	-0.2 ± 5.8
RAIL	0.5 ± 4.9
PBRL	-15.0 ± 8.2
BAYESIAN REX	1.6 ± 4.7
PG-BROIL	**23.9 ± 13.5**

(b)

Figure 4. **Atari Boxing:** We evaluate PG-BROIL against baseline imitation learning algorithms when learning from preferences over demonstrations. Results are averages (± one st. dev.) over 3 random seeds and 100 test episodes. For PG-BROIL, we set $\alpha = 0.9$ and report results for the best λ ($\lambda = 0.3$). The game score is the number of hits the trained agent (white) scored minus the number of times the agent gets hit by the opponent (black).

6. Discussion and Future Work

Summary: We derive a novel algorithm, PG-BROIL, for safe policy optimization in continuous MDPs that is robust to epistemic uncertainty over the true reward function. Experiments evaluating PG-BROIL with different prior distributions over reward hypotheses suggest that solving PG-BROIL with different values of λ can produce a family of solutions that span the Pareto frontier of policies which trade-off expected performance and robustness. Finally, we show that PG-BROIL improves upon state-of-the-art imitation learning methods when learning from small numbers of demonstrations by not just optimizing for the most likely reward function, but by also hedging against poor performance under other likely reward functions.

Future Work and Limitations: We found that PG-BROIL can sometimes become unstable for values of lambda close to zero—likely due to the indicator function in the CVaR policy gradient. We experimented with entropic risk measure (Föllmer & Knispel, 2011), a continuously differentiable alternative to CVaR, but obtained similar results to CVaR (see Appendix B). Future work also includes using contrastive learning (Laskin et al., 2020) and deep Bayesian reward function inference (Brown et al., 2020a) to enable robust policy learning from raw pixels.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful suggestions for improving the paper. This work has taken place in the AUTOLAB and InterACT Lab at the University of California, Berkeley and the Reinforcement Learning and Robustness Lab (RLsquared) at the University of New Hampshire. AUTOLAB research is supported in part by the Scalable Collaborative Human-Robot Learning (SCHooL) Project, the NSF National Robotics Initiative Award 1734633, and in part by donations from Google, Siemens, Amazon Robotics, Toyota Research Institute, and by equipment grants from NVidia. InterACT Lab research is supported in part by AFOSR, NSF NRI SCHOOL, and ONR YIP. RLsquared research is supported in part by NSF Grants IIS-1717368 and IIS-1815275. Ashwin Balakrishna is supported by an NSF GRFP. This article solely reflects the opinions and conclusions of its authors and not the views of the sponsors or their associated entities.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Achiam, J. Spinning Up in Deep Reinforcement Learning. 2018. URL `https://spinningup.openai.com/`.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31. PMLR, 2017.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Arora, S. and Doshi, P. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.

Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Brown, D., Goo, W., Nagarajan, P., and Niekum, S. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, pp. 783–792. PMLR, 2019.

Brown, D., Niekum, S., Coleman, R., and Srinivasan, R. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*, 2020a.

Brown, D., Niekum, S., and Marek, P. Bayesian robust optimization for imitation learning. In *Neural Information Processing Systems (NeurIPS)*, 2020b.

Brown, D. S. and Niekum, S. Efficient probabilistic performance bounds for inverse reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Choi, J. and Kim, K.-E. Map inference for bayesian inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1989–1997, 2011.

Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

Delage, E. and Mannor, S. Percentile optimization for markov decision processes with parameter uncertainty. *Operations research*, 58(1):203–213, 2010.

Delbaen, F. Coherent risk measures on general probability spaces. In *Advances in finance and stochastics*, pp. 1–37. Springer, 2002.

Derman, E., Mankowitz, D. J., Mann, T. A., and Mannor, S. Soft-robust actor-critic policy-gradient. *arXiv preprint arXiv:1803.04848*, 2018.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pp. 49–58. PMLR, 2016.

Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., and Tomlin, C. J. A general safety framework for learning-based control in uncertain robotic systems. In *IEEE Transactions on Automatic Control*, 2018.

Föllmer, H. and Knispel, T. Entropic risk measures: Coherence vs. convexity, model ambiguity and robust large deviations. *Stochastics and Dynamics*, 11(02n03):333–351, 2011.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. Policy shaping: integrating human feedback with reinforcement learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2625–2633, 2013.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S. J., and Dragan, A. Inverse reward design. In *Advances in neural information processing systems*, pp. 6765–6774, 2017.

Heger, M. Consideration of risk in reinforcement learning. In *Machine Learning Proceedings*, 1994.

Ho, J. and Ermon, S. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*, pp. 7461–7472, 2016.

Hoque, R., Balakrishna, A., Putterman, C., Luo, M., Brown, D. S., Seita, D., Thananjeyan, B., Novoseller, E., and Goldberg, K. Lazydagger: Reducing context switching in interactive imitation learning. *arXiv preprint arXiv:2104.00053*, 2021.

Huang, J., Wu, F., Precup, D., and Cai, Y. Learning safe policies with expert guidance. In *Advances in Neural Information Processing Systems*, pp. 9105–9114, 2018.

Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Knox, W. B. and Stone, P. Reinforcement learning from simultaneous human and mdp reward. In *AAMAS*, pp. 475–482, 2012.

Krakovna, V., Uesato, J., Mikulik, V., Rahtz, M., Everitt, T., Kumar, R., Kenton, Z., Leike, J., and Legg, S. Specification gaming examples in ai. DeepMind Blog, 2020.

Lacotte, J., Ghavamzadeh, M., Chow, Y., and Pavone, M. Risk-sensitive generative adversarial imitation learning. *arXiv preprint arXiv:1808.04468*, 2018.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Lobo, E. A., Ghavamzadeh, M., and Petrik, M. Soft-robust algorithms for handling model misspecification. *arXiv preprint arXiv:2011.14495*, 2020.

Majumdar, A., Singh, S., Mandlekar, A., and Pavone, M. Risk-sensitive inverse reinforcement learning via coherent risk models. In *Robotics: Science and Systems*, 2017.

Nass, D., Belousov, B., and Peters, J. Entropic risk measure in policy search. *arXiv preprint arXiv:1906.09090*, 2019.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., and Peters, J. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

Peters, J. and Schaal, S. Reinforcement Learning of Motor Skills with Policy Gradients. *Neural Networks*, 21(4): 682–697, 2008.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Puterman, M. L. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley-Interscience, 2005.

Ramachandran, D. and Amir, E. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pp. 2586–2591, 2007.

Ratner, E., Hadfield-Mennell, D., and Dragan, A. Simplifying reward design through divide-and-conquer. In *Robotics: Science and Systems*, 2018.

Regan, K. and Boutilier, C. Regret-based reward elicitation for Markov decision processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 444–451, 2009. ISBN 978-0-9749039-5-8.

Rockafellar, R. T., Uryasev, S., et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010.

Ross, S., Gordon, G., and Bagnell, D. 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*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Russel, R. H., Behzadian, B., and Petrik, M. Entropic risk constrained soft-robust policy optimization. *arXiv preprint arXiv:2006.11679*, 2020.

Sadigh, D., Dragan, A. D., Sastry, S., and Seshia, S. A. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

Santara, A., Naik, A., Ravindran, B., Das, D., Mudigere, D., Avancha, S., and Kaul, B. RAIL : Risk-Averse Imitation Learning Extended Abstract. *arXiv:1707.06658*, 2018.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust region policy optimization. *arXiv preprint arXiv:1707.06347*, 2017a.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

Shen, Y., Tobia, M. J., Sommer, T., and Obermayer, K. Risk-sensitive reinforcement learning. In *Neural Computation*, volume 26, 2014.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Syed, U., Bowling, M., and Schapire, R. E. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pp. 1032–1039, 2008.

Tamar, A., Glassner, Y., and Mannor, S. Policy gradients beyond expectations: Conditional value-at-risk. In *CoRR*, 2014.

Tamar, A., Glassner, Y., and Mannor, S. Optimizing the cvar via sampling. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Tang, Y. C., Zhang, J., and Salakhutdinov, R. Worst cases policy gradients. *Conf. on Robot Learning (CoRL)*, 2019.

Tang, Y. C., Zhang, J., and Salakhutdinov, R. Worst cases policy gradients. In Kaelbling, L. P., Kragic, D., and Sugiura, K. (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1078–1093. PMLR, 30 Oct–01 Nov 2020. URL http://proceedings.mlr.press/v100/tang20a.html.

Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., and Heess, N. dm_control: Software and tasks for continuous control, 2020.

Thananjeyan, B., Balakrishna, A., Rosolia, U., Gonzalez, J. E., Ames, A., and Goldberg, K. Abc-lmpc: Safe sample-based learning mpc for stochastic nonlinear dynamical systems with adjustable boundary conditions. In *Workshop on the Algorithmic Foundations of Robotics*, 2020a.

Thananjeyan, B., Balakrishna, A., Rosolia, U., Li, F., McAllister, R., Gonzalez, J. E., Levine, S., Borrelli, F., and Goldberg, K. Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks. *Robotics and Automation Letters (RAL)*, 2020b.

Thananjeyan, B., Balakrishna, A., Nair, S., Luo, M., Srinivasan, K., Hwang, M., Gonzalez, J. E., Ibarz, J., Finn, C., and Goldberg, K. Recovery rl: Safe reinforcement learning with learned recovery zones. In *Robotics and Automation Letters (RA-L)*. IEEE, 2021.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4950–4957, 2018.

Xu, K., Ratner, E., Dragan, A., Levine, S., and Finn, C. Learning a prior over intent via meta-inverse reinforcement learning. *International Conference on Machine Learning*, 2019.

Yuan, Y. Pytorch implementation of reinforcement learning algorithms. https://github.com/Khrylx/PyTorch-RL, 2019.

Zhang, J. and Cho, K. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

Zhang, S., Liu, B., and Whiteson, S. Mean-variance policy iteration for risk-averse reinforcement learning. In *Conference on Artificial Intelligence (AAAI)*, 2021.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A. Full Derivation of CVaR BROIL Policy Gradient

In this section we derive the complete derivation of the policy gradient objective for BROIL.

A.1. General Performance Metric

We will first derive a policy gradient algorithm for any performance metric. Then, we will derive special cases corresponding to particular choices of the performance metric.

We start with the same objective, which we note is a weighted combination of two terms, one of which measures expected performance ($\mathbb{E}[\psi(\pi_\theta, R)]$) and the other of which measures tail risk ($\text{CVaR}_\alpha \left[\psi(\pi_\theta, R)\right]$):

$$\underset{\pi_\theta}{\text{maximize}} \quad \lambda \cdot \mathbb{E}[\psi(\pi_\theta, R)] + (1-\lambda) \cdot \text{CVaR}_\alpha \left[\psi(\pi_\theta, R)\right] \tag{21}$$

We want to solve this via a policy gradient algorithm so we need to find the gradient with respect to θ. For the first term we have

$$\begin{aligned} \nabla_\theta \mathbb{E}_{\mathbb{P}(R)}[\psi(\pi_\theta, R)] =& \mathbb{E}_{\mathbb{P}(R)}[\nabla_\theta \psi(\pi_\theta, R)] & (22)\\ =& \sum_i \mathbb{P}(r_i) \nabla_\theta \psi(\pi_\theta, r_i). & (23) \end{aligned}$$

Now consider the gradient of the CVaR term. We have

$$\nabla_\theta \, \text{CVaR}_\alpha[\psi(\pi_\theta, R)] = \nabla_\theta \max_\sigma \Big(\sigma - \frac{1}{1-\alpha} \sum_i \mathbb{P}(r_i) \big[\sigma - \psi(\pi_\theta, r_i)\big]_+\Big) \tag{24}$$

Here we need to take the gradient with respect to an inner maximization over the auxiliary variable σ. To solve for the gradient of this term, first note that given a fixed policy π_θ, the objective is piecewise linear in σ with switch points at each sample from the posterior ($\psi(\pi_\theta, r_i)$ $\forall r_i$). Thus, we can solve for σ via linear programming or just via a line search. If we let $\psi_i = \psi(\pi_\theta, r_i)$ then we can quickly iterate over all reward function hypotheses and solve for σ as

$$\sigma^* = \underset{\sigma \in \{\psi_1, \ldots, \psi_N\}}{\text{argmax}} \Big(\sigma - \frac{1}{1-\alpha} \sum_i \mathbb{P}(r_i) \big[\sigma - \psi_i\big]_+\Big) \tag{25}$$

Given the solution to the above optimization problem, we can now fix $\sigma = \sigma^*$ and then perform a step of policy gradient optimization by following the sub-gradient of CVaR with respect to the policy parameters θ:

$$\begin{aligned} \nabla_\theta \left(\sigma^* - \frac{1}{1-\alpha} \sum_i \mathbb{P}(r_i) \big[\sigma^* - \psi(\pi_\theta, r_i)]\big]_+\right) =& -\frac{1}{1-\alpha} \sum_i \mathbb{P}(r_i) \nabla_\theta \big[\sigma^* - \psi(\pi_\theta, r_i)\big]_+ & (26)\\ =& \frac{1}{1-\alpha} \sum_i \mathbb{P}(r_i) \mathbf{1}_{\sigma^* \geq \psi(\pi_\theta, r_i)} \nabla_\theta \psi(\pi_\theta, r_i) & (27) \end{aligned}$$

where we use the notation $\mathbf{1}_x$ to denote the indicator function:

$$\mathbf{1}_x = \begin{cases} 1 & \text{if } x \text{ is True} \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

We now can formulate the full BROIL policy gradient update step by blending the policy gradient over the expectation with the policy gradient over the CVaR:

$$\begin{aligned} \nabla_\theta \text{BROIL} =& \lambda \sum_i \mathbb{P}(r_i) \nabla_\theta \psi(\pi_\theta, r_i) + \frac{1-\lambda}{1-\alpha} \sum_i \mathbb{P}(r_i) \mathbf{1}_{\sigma^* \geq \psi(\pi_\theta, r_i)} \nabla_\theta \psi(\pi_\theta, r_i) & (29)\\ =& \sum_i \mathbb{P}(r_i) \nabla_\theta \psi(\pi_\theta, r_i) \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \psi(\pi_\theta, r_i)}\right) & (30) \end{aligned}$$

A.2. Policy Gradient for Expected Return

We now consider the case where our performance metric is expected value, i.e., $\psi(\pi_\theta, R) = v(\pi_\theta, R) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$. Plugging expected value for our performance metric into Equation (29) gives the following:

$$\nabla_\theta \text{BROIL} = \sum_i \mathbb{P}(r_i) \nabla_\theta v(\pi, r_i) \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq v(\pi, r_i)} \right), \tag{31}$$

where solving for σ^* requires estimating v_i by collecting a set $\mathcal{T}$ of on-policy trajectories $\tau \sim \pi_\theta$ where $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$:

$$v_i \approx \frac{1}{|\mathcal{T}|} \sum_{\tau\in\mathcal{T}} \sum_{t=0}^{T} r_i(s_t, a_t). \tag{32}$$

Given the expected return under each reward function hypothesis we solve for σ^* as

$$\sigma^* = \underset{\sigma\in\{v_1,\ldots,v_N\}}{\text{argmax}} \left(\sigma - \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbb{P}(r_i) \big[\sigma - v_i\big]_+ \right). \tag{33}$$

Solving for σ^* does not require additional data collection beyond what is required for standard policy gradient approaches. We simply evaluate the set of rollouts $\mathcal{T}$ from π_θ under each reward function hypothesis, r_i and then solve the optimization problem above to find σ^*. While this requires more computation than a standard policy gradient approach—we have to evaluate each rollout under N reward functions—this does not increase the online data collection, which is often the bottleneck in RL algorithms.

Note that, in general, we can write the policy gradient of the expected return as

$$\nabla_\theta v(\pi, r_i) = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \right] \tag{34}$$

where $\Phi_t^{r_i}$ is some measure of the quality of the policy under reward function r_i. Common choices include the return of a trajectory: $\Phi_t^{r_i} = r_i(\tau)$, the reward-to-go from time t: $\sum_{t'=t}^{T} r_i(s_{t'}, a_{t'})$, the reward-to-go with a state-dependent baseline: $\sum_{t'=t}^{T} r_i(s_{t'}, a_{t'}) - b(s_t)$, the on-policy action-value function $Q^{\pi_\theta}(s_t, a_t)$, or the on-policy advantage function (the most popular choice) (Schulman et al., 2015):

$$\Phi_t^{r_i} = A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t). \tag{35}$$

Any of these formulations of the policy gradient can be used for the above BROIL policy gradient as follows where we approximate the expectation using a set $\mathcal{T}$ of on-policy trajectories $\tau \sim \pi_\theta$:

$$\begin{aligned}
\nabla_\theta \text{BROIL} &= \sum_i \mathbb{P}(r_i)\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)]\Big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\Big) && (36)\\
&= \sum_i \mathbb{P}(r_i)\Big(\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T}\nabla_\theta \log\pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\Big]\Big)\Big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\Big) && (37)\\
&\approx \sum_i \mathbb{P}(r_i)\Big(\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\Big[\sum_{t=0}^{T}\nabla_\theta \log\pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\Big]\Big)\Big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\Big) && (38)\\
&= \frac{1}{|\mathcal{T}|}\sum_i \mathbb{P}(r_i)\Big(\sum_{\tau\in\mathcal{T}}\Big[\sum_{t=0}^{T}\nabla_\theta \log\pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\Big]\Big)\Big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\Big) && (39)\\
&= \frac{1}{|\mathcal{T}|}\sum_i\sum_{\tau\in\mathcal{T}} \mathbb{P}(r_i)\Big[\sum_{t=0}^{T}\nabla_\theta \log\pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\Big]\Big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\Big) && (40)\\
&= \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_i \mathbb{P}(r_i)\Big[\sum_{t=0}^{T}\nabla_\theta \log\pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\Big]\Big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\Big) && (41)\\
&= \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_i\sum_{t=0}^{T} \mathbb{P}(r_i)\nabla_\theta \log\pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\Big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\Big) && (42)\\
&= \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{T}\sum_i \mathbb{P}(r_i)\nabla_\theta \log\pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\Big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\Big) && (43)\\
&= \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{T}\nabla_\theta \log\pi_\theta(a_t \mid s_t)\Big(\sum_i \mathbb{P}(r_i)\Phi_t^{r_i}(\tau)\big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\big)\Big) && (44)\\
&= \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{T}\nabla_\theta \log\pi_\theta(a_t \mid s_t)w_t && (45)
\end{aligned}$$

where

$$w_t = \sum_i \mathbb{P}(r_i)\Phi_t^{r_i}(\tau)\left(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^*\geq v(\pi,r_i)}\right) \tag{46}$$

is the weight associated with each state-action pair. Intuitively, if $\lambda = 1$, then we just focus on increasing the likelihood of actions that look good in expectation. If $\lambda = 0$, then we focus on increasing the likelihood of actions that look good under reward functions that the current policy π_θ performs poorly under, i.e., we focus on improving our performance under all r_i such that $\sigma^* > v(\pi, r_i)$), weighting the gradient according to the likelihood of these worst-case reward functions.

A.3. Policy Gradient for Baseline Regret

We now consider the case where our performance metric is baseline regret (Brown et al., 2020b), which measures performance with respect to some expert demonstrator. The intuition is that this formulation may be able to reduce variance in the policy gradient estimator by grounding updates in the expected return of the demonstrator. We define baseline regret as follows:

$$\psi(\pi_\theta, R) = v(\pi_\theta, R) - v(\pi_E, R), \tag{47}$$

where π_E denotes an expert policy and $v(\pi_E, R)$ is usually estimated from demonstrations. Plugging baseline regret for our performance metric into Equation (29) gives the following:

$$\nabla_\theta \text{BROIL} = \sum_i \mathbb{P}(r_i) \nabla_\theta \big(v(\pi_\theta, r_i) - v(\pi_E, r_i)\big) \bigg(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq v(\pi_\theta, r_i) - v(\pi_E, r_i)}\bigg) \tag{48}$$

$$= \sum_i \mathbb{P}(r_i) \nabla_\theta v(\pi_\theta, r_i) \bigg(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq v(\pi_\theta, r_i) - v(\pi_E, r_i)}\bigg) \tag{49}$$

$$= \sum_i \mathbb{P}(r_i) \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[r_i(\tau)] \bigg(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq v(\pi_\theta, r_i) - v(\pi_E, r_i)}\bigg) \tag{50}$$

In practice, we typically only have samples of expert behavior rather than a full policy. In this case, we can estimate the return of the demonstrator under reward function hypothesis r_i using a set of demonstrated trajectories $D = \{\tau_1, \dots, \tau_m\}$ as

$$v(\pi_E, r_i) \approx \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} r_i(s_t, a_t), \tag{51}$$

where T is the horizon of the demonstrations.

If r_i is a linear function, i.e.,$(s, a) = \boldsymbol{w}_i^T \phi(s, a)$, then we can compute the empirical expected feature counts using the demonstrated trajectories $D = \{\tau_1, \dots, \tau_m\}$ to get

$$\hat{\mu}_E = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{(s_t, a_t) \in \tau} \phi(s_t, a_t), \tag{52}$$

where $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^k$ denotes the reward features. We can then estimate $v(\pi_E, r_i)$ as

$$v(\pi_E, r_i) = \boldsymbol{w}_i^T \hat{\mu}_E, \tag{53}$$

where $\boldsymbol{w}_i$ is the feature weight vector corresponding to linear reward function r_i sampled from the posterior. The advantage is that we only have to evaluate the expected feature counts once and then we can use this vector to estimate the expected return under any number of reward function hypotheses via dot products.

Given the estimate baseline regret under each reward function hypothesis we solve for σ^* as

$$\sigma^* = \underset{\sigma \in \{v_1^{\text{br}}, \dots, v_N^{\text{br}}\}}{\text{argmax}} \bigg(\sigma - \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbb{P}(r_i) \big[\sigma - v_i^{\text{br}}\big]_+\bigg), \tag{54}$$

where $v_i^{\text{br}} = v(\pi_\theta, r_i) - v(\pi_E, r_i)$.

As in the previous section, if we approximate the baseline regret using a set $\mathcal{T}$ of on-policy trajectories $\tau. \sim \pi_\theta$ and a set $\mathcal{D}$ of demonstrations we have:

$$\nabla_\theta \text{BROIL} = \sum_i \mathbb{P}(r_i) \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[r_i(\tau)] \bigg(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq v_i^{\text{br}}}\bigg) \tag{55}$$

$$= \sum_i \mathbb{P}(r_i) \Bigg(\mathbb{E}_{\tau \sim \pi_\theta} \Bigg[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i}\Bigg]\Bigg) \bigg(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq v_i^{\text{br}}}\bigg) \tag{56}$$

$$\approx \sum_i \mathbb{P}(r_i) \Bigg(\frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \Bigg[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i}\Bigg]\Bigg) \bigg(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq v_i^{\text{br}}}\bigg) \tag{57}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) w_t \tag{58}$$

where

$$w_t = \sum_i \mathbb{P}(r_i)\Phi_t^{r_i}(\tau)\big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^* \geq v_i^{\text{br}}}\big) \tag{59}$$

is the weight associated with each state-action pair. The baseline regret adjusts risk such that it is riskier to explore areas of the state-space that were not visited by the demonstrator, thereby encouraging pessimism in the face of uncertainty. To see this note that

$$v_i^{\text{br}} = v(\pi_\theta, r_i) - v(\pi_E, r_i) \approx \boldsymbol{w}_i^T(\hat{\mu}_{\pi_\theta} - \hat{\mu}_E) = \sum_{j=1}^k \boldsymbol{w}_i[j](\hat{\mu}_{\pi_\theta}[j] - \hat{\mu}_E[j]), \tag{60}$$

where $\hat{\mu}_{\pi_\theta}$ are the estimated expected feature counts of π_θ and $\hat{\mu}_E$ are the estimated expected feature counts of π_E and we assume all vectors lie in $\mathbb{R}^k$. Thus, if the expert and policy both encounter reward feature j at the same frequency ($\hat{\mu}_{\pi_\theta}[j] = \hat{\mu}_E[j]$), the distribution over $\boldsymbol{w}_i[j]$ will not contribute to v_i^{br}. Thus, the tail risk will be determined by other reward weight distributions. Conversely, when there is disagreement, there will be the potential for risk: if the policy visits new states that are estimated to have negative reward weight or if the policy does not visit states visited by the demonstrator that are estimated to have positive reward weight, then either will lower v_i^{br} and result in more tail risk.

Note, however, that baseline regret does not only provide an incentive to directly imitate the demonstrator. If demonstrations are suboptimal, but we have preferences over them, (Brown et al., 2020a) demonstrated that fast Bayesian reward inference is possible. If under the posterior distribution of reward functions we have high confidence that certain states are good (positive weight) or bad (negative weight), then lower risk policies will seek to visit the bad states less often than the demonstrator and visit the good states more often. Thus, it is still possible to outperform the demonstrator while being robust to reward weights with high uncertainty by imitating to hedge against high uncertainty, but exploiting our posterior to perform better than the demonstrator when we have low uncertainty over the desirability of certain states.

B. Entropic Risk Measure Policy Gradient

Here we show that another common risk metric, Entropic Risk Measure (ERM) (Föllmer & Knispel, 2011), also is amenable to policy gradient optimization within the BROIL framework. One benefit of ERM is that it is differentiable everywhere unlike CVaR. ERM has been considered recently under the settings of risk-averse policy search under a known reward function (Nass et al., 2019) and soft-robust optimization with respect to model uncertainty (Russel et al., 2020).

B.1. Entropic Risk Measure

The entropic risk measure (Föllmer & Knispel, 2011) is another form of tail risk that has the benefit of being everywhere differentiable. The entropic risk measure (ERM) of a random variable X is defined as:

$$ERM = -\frac{1}{\alpha}\log \mathbb{E}[e^{-\alpha X}] \tag{61}$$

where $\alpha \in (0, \infty)$ represents the risk sensitivity (higher is more risk-sensitive) and where larger values of ERM indicate lower risk.

Similar to the CVaR BROIL objective we can formulate at BROIL objective using ERM. As we show in Section B.2, the policy gradient of ERM-BROIL is given by Equation 16 with

$$w_t^{ERM} = \sum_i \mathbb{P}(r_i)\Phi_t^{r_i}(\tau)\left(\lambda + (1-\lambda)\frac{e^{-\alpha v(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha v(\pi_\theta, R)}]}\right) \tag{62}$$

If $\lambda = 1$, then we just focus on increasing the likelihood of actions that look good in expectation. If $\lambda = 0$, then we focus on increasing the likelihood of actions that look good under reward functions that the current policy π_θ performs poorly under. In particular, the policy gradient for the ERM term is given by a weighted sum of policy gradients for each reward function in the posterior. The weights are softmax probabilities which will concentrate the probability around the reward function r_i for which $v(\pi_\theta, r_i)$ is lowest. Intuitively, this will encourage policy updates that improve the performance under the reward functions for which π_θ performs the worst. As $\alpha \to \infty$, the softmax probabilities will concentrate on the absolute worst-case reward in the distribution, but for $\alpha \to 0$, this probability will be distributed according to the reward function probabilities $\mathbb{P}(r_i)$ resulting in a policy gradient that seeks to maximize return under the expected reward function.

B.2. Deriviation

In this section we derive a similar policy gradient objective for BROIL that uses entropic risk measure:

$$\text{ERM}_\alpha = -\frac{1}{\alpha}\log(\mathbb{E}_R[e^{-\alpha\psi(\pi_\theta,R)}]) \tag{63}$$

We start with the objective:

$$\underset{\pi_\theta}{\text{maximize}} \quad \lambda\cdot\mathbb{E}[\psi(\pi_\theta,R)] + (1-\lambda)\cdot\text{ERM}_\alpha\Big[\psi(\pi_\theta,R)\Big] \tag{64}$$

We assume that our performance metric is expected value, i.e., $\boldsymbol{\psi}(\pi_{\boldsymbol{u}},R) = v(\pi,R) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$.

We need to find the gradient wrt θ. The first term is the same as in the previous section:

$$\nabla_\theta\cdot\mathbb{E}_{\mathbb{P}(R)}[\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]] = \sum_i \mathbb{P}(r_i)\nabla_\theta\mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)]. \tag{65}$$

Now consider the gradient of the entropic risk term. We have

$$\begin{aligned}
\nabla_\theta\text{ERM}_\alpha[v(\pi,R)] =& -\nabla_\theta\frac{1}{\alpha}\log\left(\sum_i\mathbb{P}(r_i)e^{-\alpha v(\pi_\theta,r_i)}\right) &(66)\\
=& -\frac{1}{\alpha}\frac{1}{\sum_j\mathbb{P}(R_j)e^{-\alpha v(\pi_\theta,R_j)}}\sum_i\mathbb{P}(r_i)\nabla_\theta e^{-\alpha v(\pi_\theta,r_i)} &(67)\\
=& -\frac{1}{\alpha}\frac{1}{\sum_j\mathbb{P}(R_j)e^{-\alpha v(\pi_\theta,R_j)}}\sum_i\mathbb{P}(r_i)e^{-\alpha v(\pi_\theta,r_i)}\nabla_\theta(-\alpha v(\pi_\theta,r_i)) &(68)\\
=&\sum_i\frac{\mathbb{P}(r_i)e^{-\alpha v(\pi_\theta,r_i)}}{\sum_j\mathbb{P}(R_j)e^{-\alpha v(\pi_\theta,R_j)}}\nabla_\theta v(\pi_\theta,r_i) &(69)
\end{aligned}$$

As before we will be estimating the on-policy expected return for each reward hypothesis which can be done by collecting a set $\mathcal{T}$ of trajectories $\tau\sim\pi_\theta$:

$$v(\pi_\theta,R_j) = \mathbb{E}_{\tau\sim\pi_\theta}[r_ij(\tau)] \approx \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}R_j(\tau) = \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{T}R_j(s_t,a_t). \tag{70}$$

Now we can formulate the full BROIL policy gradient update step by blending the policy gradient over the expectation with the policy gradient over the ERM:

$$\begin{aligned}
\nabla_\theta\text{BROIL} =&\lambda\sum_i\mathbb{P}(r_i)\nabla_\theta v(\pi_\theta,r_i) + (1-\lambda)\sum_i\frac{\mathbb{P}(r_i)e^{-\alpha v(\pi_\theta,r_i)}}{\sum_j\mathbb{P}(R_j)e^{-\alpha v(\pi_\theta,R_j)}}\nabla_\theta v(\pi_\theta,r_i) &(71)\\
=&\sum_i\mathbb{P}(r_i)\nabla_\theta v(\pi_\theta,r_i)\left(\lambda+(1-\lambda)\frac{e^{-\alpha v(\pi_\theta,r_i)}}{\mathbb{E}_{\mathbb{P}(R)}[e^{-\alpha v(\pi_\theta,R)}]}\right) &(72)
\end{aligned}$$

As before we can write the policy gradient as

$$\nabla_\theta v(\pi_\theta,r_i) = \nabla_\theta\mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\Phi_t^{r_i}\right]. \tag{73}$$

Defining $\Phi_t^{r_i}$ in terms of a particular reward function hypothesis r_i and approximating expectations with a set $\mathcal{T}$ of on-policy trajectories $\tau \sim \pi_\theta$ gives:

$$\nabla_\theta \text{BROIL} \approx \sum_i \mathbb{P}(r_i)\left(\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\right]\right)\left(\lambda + (1-\lambda)\frac{e^{-\alpha v(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha v(\pi_\theta, R)}]}\right) \tag{74}$$

$$=\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{T}\sum_i \mathbb{P}(r_i)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Phi_t^{r_i}\left(\lambda + (1-\lambda)\frac{e^{-\alpha v(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha v(\pi_\theta, R)}]}\right) \tag{75}$$

$$=\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(\sum_i \mathbb{P}(r_i)\Phi_t^{r_i}(\tau)\left(\lambda + (1-\lambda)\frac{e^{-\alpha v(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha v(\pi_\theta, R)}]}\right)\right) \tag{76}$$

$$=\frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t \mid s_t) w_t \tag{77}$$

where

$$w_t = \sum_i \mathbb{P}(r_i)\Phi_t^{r_i}(\tau)\left(\lambda + (1-\lambda)\frac{e^{-\alpha v(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha v(\pi_\theta, R)}]}\right) \tag{78}$$

is the weight associated with each state-action pair. Intuitively, if $\lambda = 1$, then we just focus on increasing the likelihood of actions that look good in expectation. If $\lambda = 0$, then we focus on increasing the likelihood of actions that look good under reward functions that the current policy π_θ performs poorly under.

B.3. Experiments

CartPole Using the same posterior and same hyperparameters as the original experiment, we redo the experiment except using ERM as the risk metric. Figure 5 shows the tradeoff between robustness and expected return for various λ. We find that results with the ERM risk metric are relatively similar to those with the CVaR risk metric in the main text.

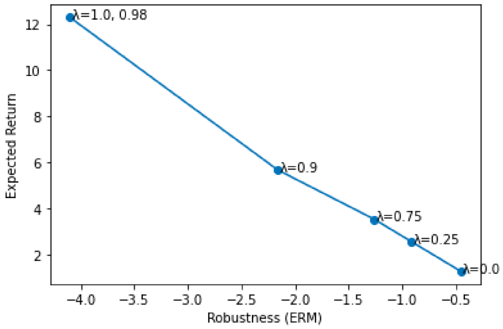


Figure 5. Efficient frontier curve for CartPole with the ERM risk metric. We set α equal to 0.001 and test over multiple values of λ. PG-BROIL with ERM acheives similar stability to CVaR.

Pointmass Navigation Figure 6 shows the Pointmass Navigation task with the ERM risk measure. Overall, the behavior is very similar. One distinction is that for lower values of lambda (ie $0, 0.2$) the pointmass goes through the edge of the gray region while for CVaR the pointmass avoided the gray region entirely.

TrashBot Figure 7 shows the same TrashBot experiment with the ERM risk metric and $\alpha = 1$. We find that results are similar when ERM is used instead of CVaR. With CVaR we saw $\lambda = 0.8$ gave the best results while for ERM $\lambda = 0.7$ was best.

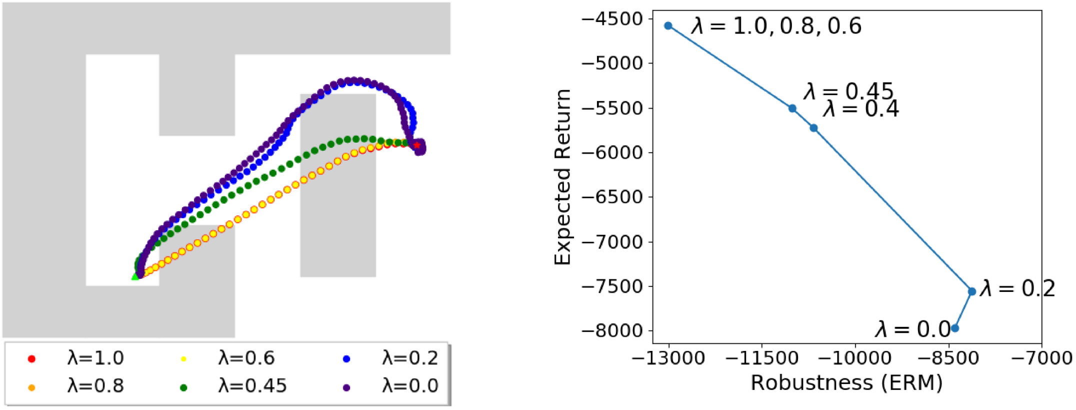


Figure 6. We show qualitative performance (left) and an efficient frontier curve (right) for the same environment and parameters as Figure 2b, but use ERM as the risk measure instead of CVaR for different values of lambda.

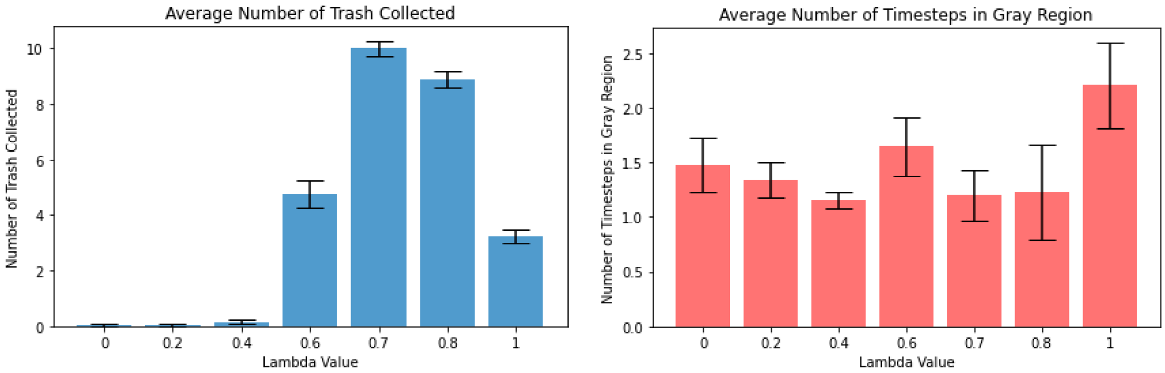


Figure 7. We run the TrashBot environment with the ERM risk metric and $\alpha = 1$ over various values of λ. We take the 95% confidence interval and plot them as the error bars. We find that the TrashBot collects the most trash when $\lambda = 0.7$ while minimizing the amount of time in the grey region.

C. Trust Region PG-BROIL

We now derive a version of the Proximal Policy Optimization (PPO) (Schulman et al., 2017b) algorithm for optimizing the BROIL objective. We specifically consider the PPO-clip objective, which adjusts the advantage function to encourage controlled updates of the policy at each epoch. Precisely, let the policy parameters at epoch k be given by θ_k. Then PPO-clip implements the following update:

$$\theta_{k+1} = \operatorname*{argmax}_{\theta} \mathbb{E}_{(s,a)\sim\pi_{\theta_k}}[L(a, s, \theta_k, \theta)] \tag{79}$$

where

$$L(a, s, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), g(\epsilon, A^{\pi_{\theta_k}}(s,a))\right) \tag{80}$$

and

$$g(\epsilon, A^{\pi_{\theta_k}}(s,a)) = \begin{cases} (1+\epsilon)A^{\pi_{\theta_k}}(s,a) & A^{\pi_{\theta_k}}(s,a) \geq 0 \\ (1-\epsilon)A^{\pi_{\theta_k}}(s,a) & A^{\pi_{\theta_k}}(s,a) < 0 \end{cases} \tag{81}$$

To implement a PPO-style gradient clipping for PG-BROIL, we replace $A^{\pi_{\theta_k}}(s,a)$ with the BROIL Policy Gradient weights:

$$w_t = \sum_i \mathbb{P}(r_i)\Phi_t^{r_i}(\tau)\big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^* \geq v(\pi, r_i)}\big) \tag{82}$$

where w_t is the weight associated with each state-action pair.

The full PPO-clip objective for BROIL is shown in Algorithm 2.

Algorithm 2 PPO-clip BROIL

```
Input: initial policy parameters $\theta_0$, samples from reward function posterior $R_1, \dots, R_N$ and associated probabilities, $\mathbb{P}(R_1), \dots, \mathbb{P}(R_N)$, and any form for policy gradient weights $\Phi_t$
for $k = 0, 1, 2, \dots$ do
    Collect set of trajectories $\mathcal{T}_k = \{\tau_i\}$ by running policy $\pi_\theta$ in the environment.
    Estimate expected return of $\pi_\theta$ under each reward function hypothesis $r_j$ using Eq. (12).
    Solve for $\sigma^*$ using Eq. (11)
    Update $\theta$ with stochastic gradient ascent by maximizing the PPO-clip objective:
        $\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \Big[ \frac{1}{T} \sum_{t=0}^{T} \min \Big( \frac{\pi_\theta(a|s)}{\pi_{\theta k}(a|s)} w_t, g(\epsilon, w_t) \Big) \Big]$
    using Eq. (82) for $w_t$.
end for
```

D. Experiment Hyperparameters and Details

The hyperparameters used for PPO are in Table 3, unless otherwise specified in the experiment's individual section.

D.1. Cart Pole

We modify the Open AI Gym Cartpole environment (Brockman et al., 2016) but modify the reward function to be a linear function of the cart's position by taking the cart position and multiplying it by -1, -0.8, -0.6, -0.4, -0.2, 0, and 0.2 to get our multiple reward hypotheses. For policy optimization, we implement PG-BROIL on top of the REINFORCE implementation from (Achiam, 2018) with all parameters set to their default settings except for $\alpha = 0.95$ and epochs set to 100.

D.2. Pointmass Navigation

We build on the pointmass navigation environment from (Thananjeyan et al., 2020b) and construct a system in which a pointmass agent navigates from a fixed start state to a fixed goal state with linear Gaussian dynamics. The agent can exert force in cardinal directions and experiences drag coefficient ψ and Gaussian process noise $z_t \sim \mathcal{N}(0, \sigma^2 I)$ in the dynamics. We utilize $\psi = 0.2$ and $\sigma = 0.05$ for all experiments. We include gray regions of uncertain cost as specified in the main text. For policy optimization, we implement PG-BROIL on top of the PPO implementation from (Achiam, 2018) with all

Table 3. PG-BROIL hyperparameters when built on PPO.

Hyperparameter	Value
Clip Ratio	0.2
Environment Steps per Epoch	4000
GAE Lambda	0.95
Gamma	0.99
Hidden Units	64
Network Layers	2
Optimizer	Adam
Policy Learning Rate	2e-4
Target KL	0.01
Value Learning Rate	1e-3

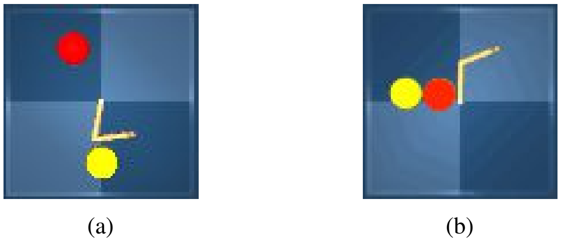


Figure 8. Reacher environment during demonstration time (a) and policy training time (b). During demonstrations, the uncertain region (red) is far from the robot arm and the goal (yellow), but during policy optimization the goal position is randomized and sometimes the uncertain cost region is in the way forcing the agent to either go around or through it.

parameters set to their default settings except for $\alpha = 0.96$, policy learning rate set to 3e-4, and epochs set to 50.

D.3. Reacher

We build on the Reacher implementation from the DeepMind Control Suite (Tassa et al., 2020) by adding a region with uncertain cost as specified in the main text. For policy optimization, we implement PG-BROIL on top of the PPO implementation from (Achiam, 2018) with all parameters set to their default settings except for $\alpha = 0.9$, policy learning rate set to 1e-4, hidden units set to 128, and epochs set to 800. To obtain preferences over the demonstrations, we rank each demonstration by the ground truth reward and assign pairwise preferences between each adjacent pair. Demonstrations were obtained by training a Soft Actor-Critic agent (Haarnoja et al., 2018) for 100 episodes and check-pointing the policy at each episode during training. This gives 100 demonstrations, and of these six with sufficiently different rewards were sampled.

D.4. TrashBot

The TrashBot dynamics and actions are the same as in the Pointmass Navigation environment except that the system dynamics are deterministic. For policy optimization, we implement PG-BROIL on top of the PPO implementation from (Achiam, 2018) with all parameters set to their default settings except for $\alpha = 0.95$, policy learning rate set to 3e-4, and epochs set to 50.

D.5. Atari Boxing

The Atari Boxing hyperparameters are the same as described in 3 with $\alpha = 0.9$ and $\lambda = 0.3$ for PG-BROIL. We use a PG-BROIL implementation on top of the PPO implementation from (Achiam, 2018) with the default hyperparameters and epochs set to 800. To obtain preferences over the demonstrations, we rank each demonstration by its game score and assign pairwise preferences between each adjacent pair. Demonstrations were obtained by training a PPO agent with the standard hyperparameters in Table 3 for 5 epochs and then taking four rollouts of episodes from the model.

E. Baseline Algorithm Details

PBRL We implement PBRL by using the pairwise preference learning loss considered in (Christiano et al., 2017). We consider learning from offline preferences and build on the implementation from (Brown et al., 2019). MCMC was performed for 20,000 steps with a proposal step size of 0.5. Weights are normalized so that $\|w\|_1 = 1$.

Bayesian REX We utilize the Bayesian REX implementation from (Brown et al., 2020b) to learn a Bayesian posterior over reward functions from offline preferences. MCMC was also performed for 20,000 sample steps with a proposal step size of 0.5. Weights are normalized so that $\|w\|_1 = 1$. We utilize a burn-in of 500 sample steps and down-sample to 20 samples.

GAIL We utilize the GAIL implementation from (Yuan, 2019). We utilize PPO for policy optimization and use most of the default parameters from the provided implementation in (Yuan, 2019). The only default parameters we changed were the L2 regularization coefficient for the weights of the discriminator network (set to $1e-2$), log std for the policy (set to -0.5), the hidden units of the policy network (set to 64), and the total number of environment steps which we varied through a

Table 4. We run GAIL with differing number of environment steps and then compare PG-BROIL with GAIL with the same number of steps. Table 1 contains both GAIL and PG-BROIL with 2×10^5 steps. Results are averages (± one st. dev.) over 100 test episodes each with a horizon of 100 steps per episode.

Algorithm	Number of Environment Steps (x10^5)	Avg. Trash Collected	Avg. Steps in Gray Region
GAIL	164	3.32 ± 1.66	0.35 ± 1.79
GAIL	41	2.88 ± 1.66	3.73 ± 7.98
GAIL	**2**	2.27 ± 1.66	5.08 ± 13.01
PG-BROIL	**2**	9.20 ± 2.19	2.04 ± 5.94

combination of changing the number of steps between each discriminator/policy update and the number of total iterations. We varied the number of environment steps and noted the behavior in Table 4. We found that on TrashBot with orders of magnitude more environmental steps we could not get consistently better performance across both trash collected and steps in the gray region so we report the performance with an equivalent number of environmental steps to PG-BROIL for all experiments.

BC We utilize the same stochastic policy and learning rate scheduler as for PPO but simply maximize the log-likelihood of actions in each of the states in the demonstrations. The learning rate for the policy is $1e-2$ and the number of BC iterations is 1000.

F. TrashBot Further Analysis and Visualization

F.0.1. Example Rollouts

In Figures 9-13 we show both successful and unsuccessful rollouts from fully trained policies for PG-BROIL and all baselines to gain intuition for their quantitative performance. In all rollouts below, on the left we show a successful case while the middle and right images are failure cases. As noted in the experiments section in the main text, PBRL places a small positive weight on staying in the white region, resulting in it falling in a local minima where it mostly optimizes for staying in the white region rather than collecting trash. This leads to low visitation of the gray region as desired, but relatively inconsistent performance in picking up pieces of trash. Bayesian REX on the other hand weights picking up trash and staying in the white region roughly equally. Thus, Bayesian REX explores the entire white region, not just the central portion where the trash is located, resulting in frequent forays into the gray region. PG-BROIL is able to successfully pick up trash and avoid excessive steps in the gray region by hedging against all reward hypotheses with sufficient probability, allowing it to recognize that it is more important to collect trash than simply stay in the white region.

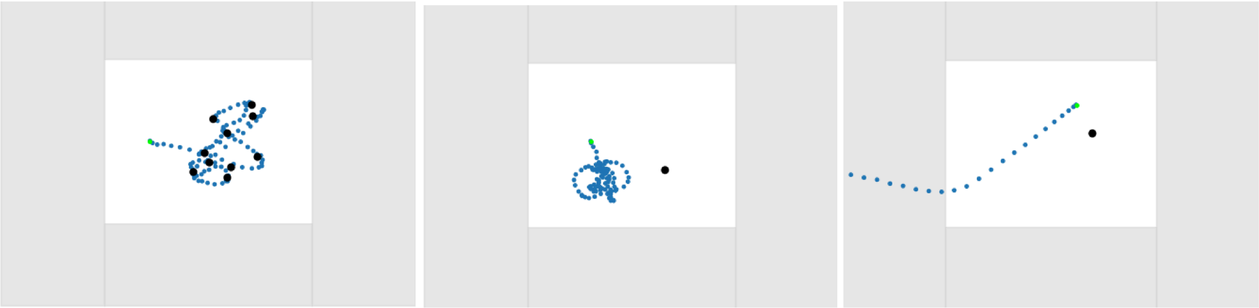


Figure 9. **PG-BROIL:** The left and middle images show trajectories for PG-BROIL with lambda value of 0.8 while the right image shows a failure case for lambda value of 0.7. PG-BROIL is able to successfully pick up trash and avoid excessive steps in the gray region by hedging against all reward hypotheses with sufficient probability, allowing it to recognize that it is more important to collect trash than simply stay in the white region.

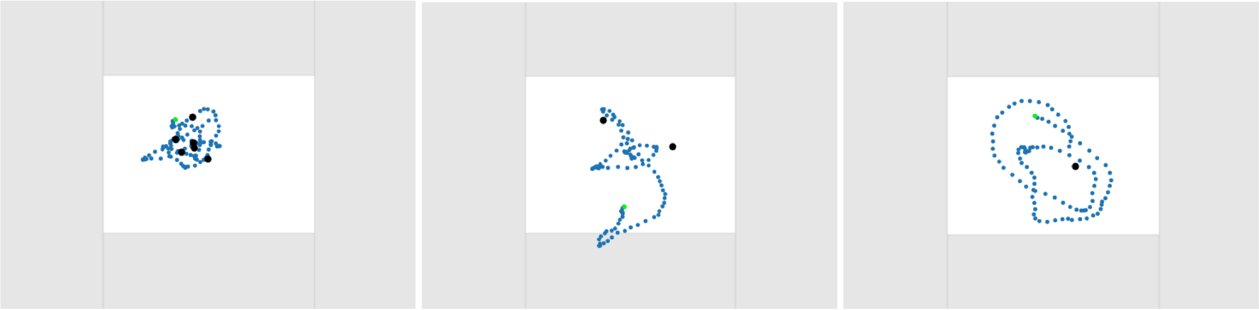


Figure 10. **PBRL:** PBRL places a small positive weight on staying in the white region, resulting in it falling in a local minima where it mostly optimizes for staying in the white region rather than collecting trash. This leads to low visitation of the gray region as desired, but relatively inconsistent performance in picking up pieces of trash.

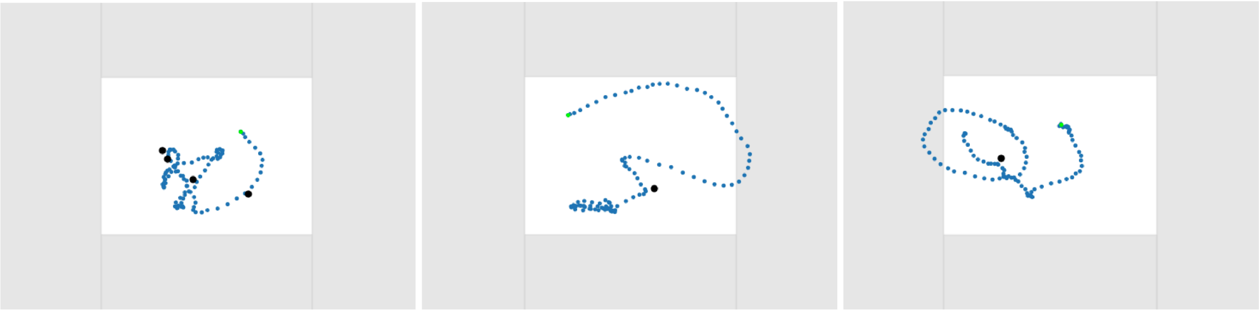


Figure 11. **Bayesian REX:** Bayesian REX weights picking up trash and staying in the white region roughly equally. Thus, Bayesian REX explores the entire white region, not just the central portion where the trash is located, resulting in frequent forays into the gray region.

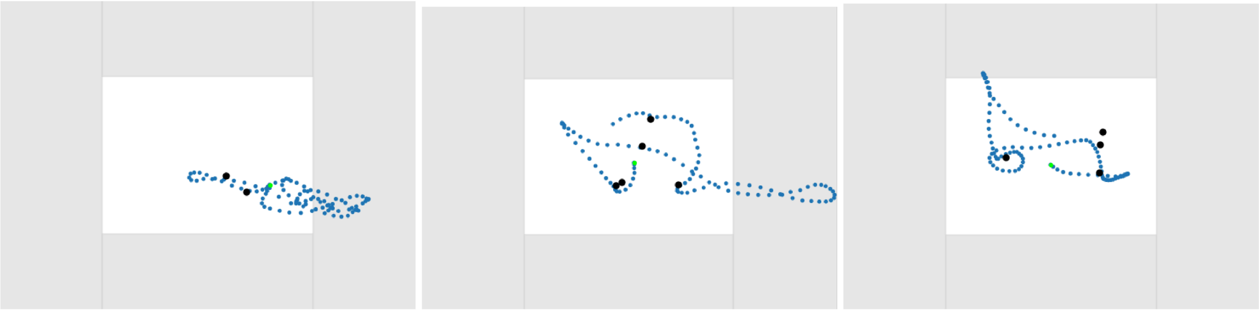


Figure 12. **GAIL:** The left, middle and right images show an example trajectory from GAIL running with 2×10^5, 4.1×10^6, and 1.64×10^7 environment steps respectively. Due to the lack of environment steps, the bot in the left and middle images take more steps in the gray region before turning around and going back to the white region. However, the bot in right image immediately turns around as soon as it contacts the gray region. The bot in the middle and right images also collect more trash in their episodes than the left image. This behavior is consistent with the averages in Table 4.

F.0.2. Posterior Analysis

Figure 14 shows the distribution of the weights for each feature for PG-BROIL. PG-BROIL exploits the fact that some reward functions have a negative weight for the WHITE feature to recognize that simply staying in the white region without going for trash is a highly suboptimal strategy. This allows PG-BROIL to outperform PBRL, which falls into a local maxima by simply mining rewards by staying in the white region.

Additionally, amongst the 20 reward functions generated on seed 0, the WHITE and TRASH features have a Pearson correlation coefficient of -0.46. This implies that if a reward function places high weight on the WHITE feature, it is likely to place a smaller or more negative weight on the TRASH feature and vice-versa. This helps create the causal confusion we

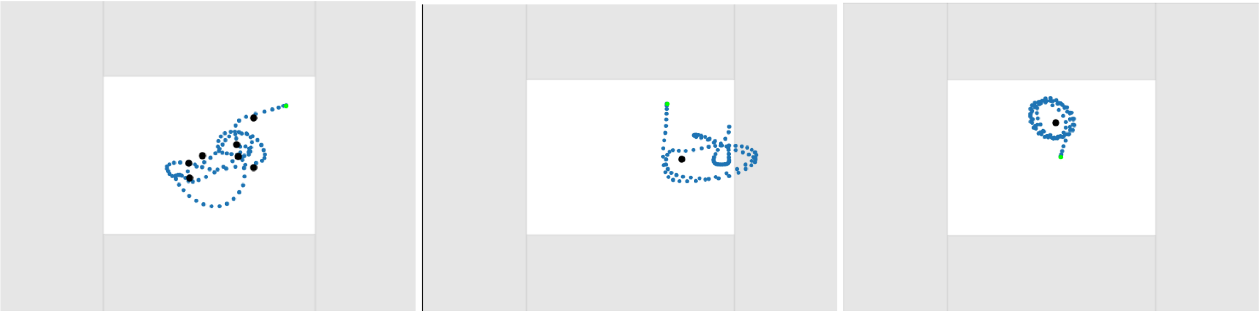


Figure 13. **BC:** The failure cases come from one of the demonstrations having the same behavior of circling the trash without picking it up. Since BC is only incentivized to exactly mimic the actions in demonstration states, it is unable to navigate ambiguities in the demos.

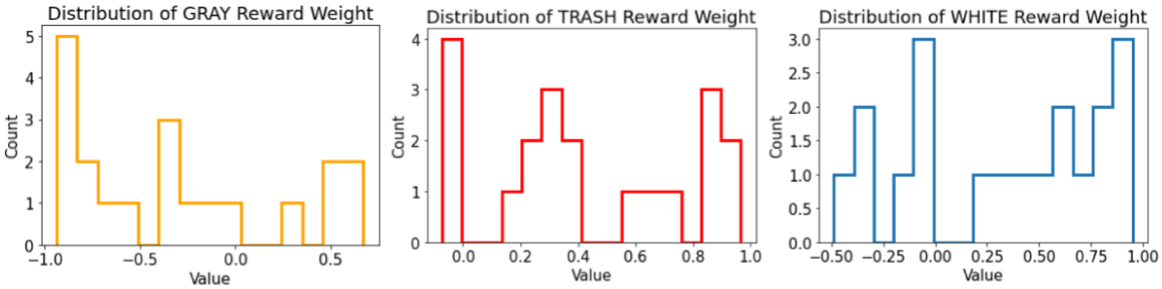


Figure 14. Distribution of each feature weight in posterior for seed 0.

see in this experiment since it is unclear whether the agent should be rewarded more for the WHITE feature or the TRASH feature.

F.0.3. Sensitivity to λ

Figure 15 shows the TrashBot experiment results over various values of λ. We found $\lambda = 0.8$ to give the best performance in terms of trash collection and gray space avoidance.

G. Sensitivity to Alpha

Most applications of CVaR use $\alpha \in [0.9, 1)$ since as $\alpha \to 0$ CVaR is equivalent to expected value. Empirically, we found that $\alpha > 0.8$ is required to get behaviors different from those that simply maximize expected reward, i.e., λ has little to no effect on the resulting policy behavior for $\alpha \leq 0.8$.

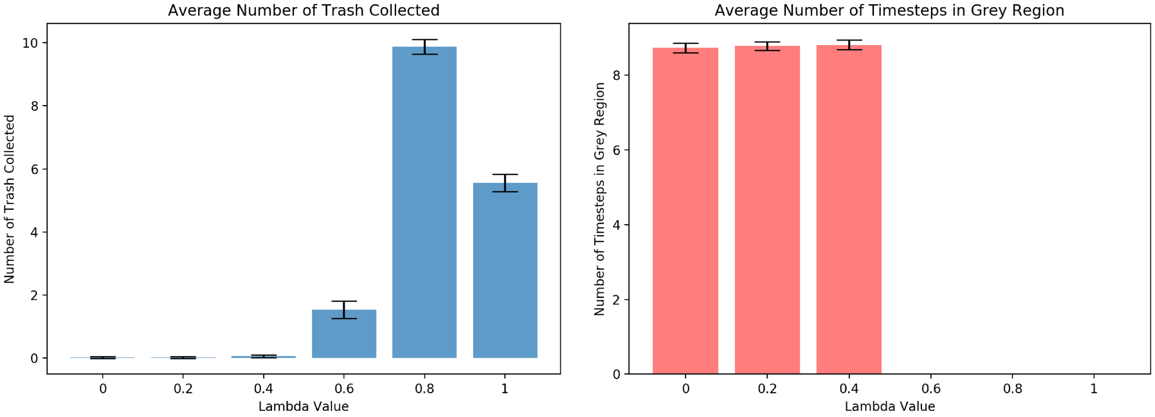


Figure 15. We run the TrashBot environment with CVaR risk metric and $\alpha = 0.95$ over various λ. We take the 95% confidence interval and plot them as the error bars. We find that the TrashBot collects the most trash with the minimum amount of timesteps spent in the grey region when $\lambda = 0.8$.