arXiv:2007.12315v3 [cs.LG] 8 Nov 2020

Bayesian Robust Optimization for Imitation Learning

Daniel S. Brown*
UC Berkeley
dsbrown@berkeley.edu

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

Marek Petrik
University of New Hampshire
mpetrik@cs.unh.edu

Abstract

One of the main challenges in imitation learning is determining what action an agent should take when outside the state distribution of the demonstrations. Inverse reinforcement learning (IRL) can enable generalization to new states by learning a parameterized reward function, but these approaches still face uncertainty over the true reward function and corresponding optimal policy. Existing safe imitation learning approaches based on IRL deal with this uncertainty using a maxmin framework that optimizes a policy under the assumption of an adversarial reward function, whereas risk-neutral IRL approaches either optimize a policy for the mean or MAP reward function. While completely ignoring risk can lead to overly aggressive and unsafe policies, optimizing in a fully adversarial sense is also problematic as it can lead to overly conservative policies that perform poorly in practice. To provide a bridge between these two extremes, we propose Bayesian Robust Optimization for Imitation Learning (BROIL). BROIL leverages Bayesian reward function inference and a user specific risk tolerance to efficiently optimize a robust policy that balances expected return and conditional value at risk. Our empirical results show that BROIL provides a natural way to interpolate between return-maximizing and risk-minimizing behaviors and outperforms existing risk-sensitive and risk-neutral inverse reinforcement learning algorithms. Code is available at https://github.com/dsbrown1331/broil.

1 Introduction

Imitation learning [42] aims to train an agent without hand-specifying a reward function by providing demonstrations. One of the main challenges in imitation learning is determining what action an agent should take when outside the states contained in the demonstrations. Inverse reinforcement learning (IRL) [40] is an approach to imitation learning in which the learning agent seeks to recover the reward function of the demonstrator. Learning a parameterized reward function provides a compact representation of the demonstrator's preferences and enables generalization to new states unseen in the demonstrations via policy optimization. However, IRL approaches still result in uncertainty over the true reward function and this uncertainty can have negative consequences if the learning agent infers a reward function that leads it to learn an incorrect policy. In this paper we propose that an imitation learning agent should learn a policy that is robust with respect to its uncertainty over the true objective of a task, but also be able to effectively trade-off epistemic risk with expected return.

For example, consider two scenarios: (1) an autonomous car detects a novel object lying in the road ahead of the car and (2) a domestic service robot tasked with vacuuming encounters a pattern on the floor it has never seen before. The first example concerns autonomous driving where the car's decisions have potentially catastrophic consequences. Thus, the car should treat the novel object as a hazard and either slow down or safely change lanes to avoid running into it. In the second example, vacuuming the floors of a house has certain risks, but the consequences of optimizing the wrong

*Work done while at UT Austin.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

reward function are arguably much less significant. Thus, when the vacuuming robot encounters a novel floor pattern it does not need to worry as much about negative side-effects.

Risk-averse optimization, especially in financial domains, has a long history of seeking to address the trade-off between risk and return using measures of risk such as variance [39], value at risk [32] and conditional value at risk [50]. This work has been extended to risk-averse optimization in Markov decision processes [16, 45, 46] and in the context of reinforcement learning [25, 60, 61], where the transition dynamics and reward function are not known. However, there has only been limited work in applying techniques for trading off risk and return in the domain of imitation learning. Brown et al. [12] seek to bound the value at risk of a policy in the imitation learning setting; however, directly optimizing a policy for value at risk is NP-hard [18]. Lacotte et al. [35] and Majumadar et al. [37] assume that risk-sensitive trajectories are available from a safe demonstrator and seek to optimize a policy that matches the risk-profile of this expert. In contrast, our approach directly optimizes a policy that balances expected return and conditional value at risk [50] which can be done via convex optimization. Furthermore, we do not try to match the demonstrator's risk sensitivity, but instead find a robust policy with respect to uncertainty over the demonstrator's reward function, allowing us to optimize policies that are potentially safer than the demonstrations.

One of the concerns of imitation learning, and especially inverse reinforcement learning, is the possibility of learning an incorrect reward function that leads to negative side-effects, for example, a vacuuming robot that learns that it is good to vacuum up dirt, but then goes around making messes for itself to clean up [52]. To address negative side-effects, most prior work on safe inverse reinforcement learning takes a minmax approach and seeks to optimize a policy with respect to the worst-case reward function [27, 30, 59]; however, treating the world as if it is completely adversarial (e.g., completely avoiding a novel patch of red flooring because it could potentially be lava [27]) can lead to overly conservative behaviors. On the other hand, other work on inverse reinforcement learning and imitation learning takes a risk neutral approach and simply seeks to perform well in expectation with respect to uncertainty over the demonstrator's reward function [48, 67]. This can result in behaviors that are overly optimistic in the face of uncertainty and can lead to policies with high variance in performance which is undesirable in high-risk domains like medicine or autonomous driving. Instead of assuming either a purely adversarial environment or a risk-neutral one, we propose the first inverse reinforcement learning algorithm capable of appropriately balancing caution with expected performance in a way that reflects the risk-sensitivity of the particular application.

The main contributions of this work are: (1) We propose Bayesian Robust Optimization for Imitation Learning (BROIL), the first imitation learning framework to directly optimize a policy that balances the expected return and the conditional value at risk under an uncertain reward function; (2) We derive an efficient linear programming formulation to compute the BROIL optimal policy; (3) We propose and compare two instantiations of BROIL: optimizing a purely robust policy with respect to uncertainty and optimizing a policy that minimizes baseline regret with respect to expert demonstrations; and (4) We demonstrate that BROIL achieves better expected return and robustness than existing risk-sensitive and risk-neutral IRL algorithms, as well as providing a richer class of solutions that correctly balance performance and risk based on different levels of risk aversion.

2 Related Work

An important challenge in inverse reinforcement learning (IRL) is dealing with ambiguity over the reward function [40, 67], since there are usually an infinite number of reward functions that are consistent with a set of demonstrations [40]. Problems with such ambiguous parameters can be solved using *robust optimization* techniques, which compute the best policy for the worst rewards consistent with the demonstrations [7]. Indeed, many IRL methods optimize policies for the worst-case rewards [27, 29, 30, 59]. This optimization for the worst-case parameter values is well known to lead to overly conservative solutions across many domains [18, 31, 51]. Bayesian IRL infers a posterior distribution over which rewards are most likely, but often only optimize for the mean [48] or MAP [15] reward function. Instead of optimizing only for the mean or worst-case reward values, we optimize for the expected performance across uncertain rewards while ensuring acceptable performance with high confidence. We rely on coherent measures of risk to represent the trade-off between the average and worst-case performance [4, 23, 56]. Similar approaches to parameter uncertainty, also known as epistemic uncertainty, have been referred to as soft-robustness in earlier work [6, 20] but have not been studied in the context of IRL.

Table 1: Summary of differences between BROIL and related robust IRL algorithms.

	BROIL (ours)	RS-GAIL [35]	VaR-BIRL [11, 12]	RBIRL [66]	FPL-IRL [30]	LPAL [59]	GAIL [29]
Bayesian robust criterion	✓	·	✓	·	·	·	·
Risk-averse expert	·	✓	·	·	·	·	·
Exploits Bayesian prior	✓	·	✓	✓	·	·	·
Robust to bad demos	·	·	·	✓	·	·	·
Baseline regret objective	✓	✓	✓	·	·	✓	✓
Optimizes policies	✓	✓	·	✓	✓	✓	✓

In Table 1, we summarize pertinent properties of IRL methods with a focus on robustness or risk-aversion. VaR-BIRL [11, 12], a closely-related method, uses VaR, a risk measure, to quantify the robustness of a given policy in Bayesian IRL. Unfortunately, extending VaR-BIRL to policy optimization is difficult since the problem of optimizing VaR in an MDP with uncertain rewards is NP-hard [18]. Additionally, VaR ignores both the tail-risk of the distribution, as well as its average value, which may be undesirable for highly risk-sensitive problems [50].

While some of the methods in Table 1 resemble our approach, they differ either in their focus or the approach. RS-GAIL and related algorithms [35, 37, 54] also mitigate risk in IRL but assume risk-averse experts and focus on optimizing policies that match the risk-aversion of the demonstrator. These methods focus on the uncertainty induced by transition probabilities, also known as aleatoric risk. The challenges in this area are very different and there is no obvious way to adapt risk-averse IRL to our Bayesian robust setting where we seek to be robust to epistemic risk rather than seeking to match the risk of the demonstrator. RBIRL [66] aims to infer a posterior distribution that is robust to small numbers of bad demonstrations, but does not address robust policy optimization with respect to ambiguity in the learned posterior. While not explicitly robust to bad demonstrations, our method makes use of any posterior distribution over reward functions and can easily be extended to use posteriors generated from methods like RBIRL [66]. Finally, FPL-IRL [30], LPAL [59], and GAIL [29] optimize the policy for a (regularized) worst-case realization of the rewards and do not attempt to balance it with the average performance.

Another important point of difference among robust IRL algorithms is the objectives they optimize for. For example, FPL-IRL [30] focuses on the absolute performance of a policy, while GAIL [24, 29] optimizes the regret (or loss) relative to the policy of the demonstrator. In reinforcement learning, it has been shown that optimizing the regret is more appropriate if a good baseline policy is available [34, 36, 45]. There are similar advantages to optimizing the regret for the optimal policy [2, 3, 49]. We would also like to emphasize that our setting is quite different from robust RL methods which focus on uncertain transition probabilities rather than rewards [22, 28, 51, 62, 64]. Unlike much robust RL work, the optimization problems we derive are tractable without requiring rectangularity assumptions [26, 38].

3 Preliminaries

Before describing our method in Section 4, we briefly introduce our notation and review some of the concepts necessary to understand our approach. We use uppercase boldface and lowercase boldface characters to denote matrices and vectors respectively.

3.1 Markov Decision Processes

We model the environment as a Markov Decision Process (MDP) [47]. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, r, P, \gamma, p_0)$, where $\mathcal{S} = \{s_1, \ldots, s_S\}$ are the states, $\mathcal{A} = \{a_1, \ldots, a_A\}$ are the actions, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition function, $\gamma \in [0, 1)$ is the discount factor, and $p_0 \in \Delta^S$ is the initial state distribution with Δ^k denoting the probability simplex in k-dimensions.

A policy is denoted by $\pi : \mathcal{S} \to \Delta^A$. When learning from demonstrations, we denote the expert's policy by $\pi_E : \mathcal{S} \to \mathcal{A}$. The rewards received by a policy at each state are $\boldsymbol{r}_\pi$ where $\boldsymbol{r}_\pi(s) = \mathbb{E}_{a \sim \pi(s)}[r(s, a)]$ and the transition probabilities for a policy π: $\boldsymbol{P}_\pi$, treated as a ma-

trix, are defined as: $\boldsymbol{P}_\pi(s, s') = \mathbb{E}_{a\sim\pi(s)}[P(s, a, s')] = \sum_a \pi(a \mid s)P(s, a, s')$. We denote the state-action occupancies of policy π as $\boldsymbol{u}_\pi \in \mathbb{R}^{S\cdot A}$, where $\boldsymbol{u}_\pi = ({\boldsymbol{u}_\pi^{a_1}}^\mathsf{T}, \ldots, {\boldsymbol{u}_\pi^{a_A}}^\mathsf{T})^\mathsf{T}$ and $\boldsymbol{u}_\pi^a(s) = \mathbb{E}[\sum_{t=0}^\infty \gamma^t \cdot \mathbf{1}_{(s_t=s\wedge a_t=a)}]$. If we denote the reward function as a vector $\boldsymbol{r} \in \mathbb{R}^{S\cdot A}$, with $\boldsymbol{r} = (r(s_1, a_1), r(s_2, a_1), \ldots, r(s_S, a_1), r(s_1, a_2), \ldots, r(s_S, a_A))^\mathsf{T}$, then the expected return of policy π under the reward function r is denoted by $\rho(\pi, r) = \boldsymbol{u}_\pi^\mathsf{T}\boldsymbol{r}$.

Linear Reward Functions We assume, without loss of generality, that the reward function $\boldsymbol{r} \in \mathbb{R}^{S\cdot A}$ can be approximated as a linear combination of k features $\boldsymbol{r} = \boldsymbol{\Phi}\boldsymbol{w}$, where $\boldsymbol{\Phi} \in \mathbb{R}^{S\cdot A\times k}$ is the linear feature matrix with rows as states and columns as features and $\boldsymbol{w} \in \mathbb{R}^k$. If $\boldsymbol{\Phi}$ is the identity matrix, then each state-action pair is allowed a unique reward. However, it is often the case that the rewards at different states are correlated via observable features which can be encoded in $\boldsymbol{\Phi}$. Note that the assumption of a linear reward function is not necessarily restrictive as these features can be arbitrarily complex nonlinear functions of the state and could be obtained via unsupervised learning from raw state observations [13, 14, 57]. Given $\boldsymbol{r} = \boldsymbol{\Phi}\boldsymbol{w}$, we denote the expected discounted feature counts of a policy as $\boldsymbol{\mu}_\pi = \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{u}_\pi$, where $\boldsymbol{\mu}_\pi \in \mathbb{R}^k$. In this case, the return of a policy is given by $\rho(\pi, r) = \boldsymbol{u}_\pi^\mathsf{T}\boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{\mu}_\pi^\mathsf{T}\boldsymbol{w}$.

Distributions over Reward Functions We are interested in problems where there is uncertainty over the true reward function r. We will model this uncertainty as a distribution over R, the random variable representing the true reward function. This distribution could be a prior distribution $\mathbb{P}(R)$ that the agent has learned from previous tasks [65]. Alternatively the distribution could be the posterior distribution $\mathbb{P}(R \mid D)$ learned via Bayesian inverse reinforcement learning [48] given demonstrations D or the posterior distribution $\mathbb{P}(R \mid R')$ learned via inverse reward design given a human-specified proxy reward function R' [27]. While the distribution over R may have an analytic form, this distribution is typically only available via sampling techniques such as Markov chain Monte Carlo (MCMC) sampling [12, 27, 48]. When there are no good priors for R, one may resort to Bayesian modeling techniques that mitigate the negative impacts of misspecified priors (e.g., [8]).

3.2 Risk Measures

Value at Risk When dealing with measures of risk, we assume that lower values are worse. Thus, as depicted in Figure 1, we want to maximize the value at risk (VaR) or conditional value at risk (CVaR). Given a risk-aversion parameter $\alpha \in [0, 1]$, the VaR_α is the $(1-\alpha)$-quantile worst-case outcome. Thus, VaR_α can be written as $\mathrm{VaR}_\alpha[X] = \sup\{x : \mathbb{P}(X \geq x) \geq \alpha\}$. Typical values of α for risk-sensitive applications are $\alpha \in [0.9, 1]$. Despite the popularity of VaR, optimizing a policy for VaR has several problems: (1) VaR is not convex and leads to an NP hard optimization problem [18], (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 catastrophic outcomes, and (3) VaR is not a coherent measure [4].

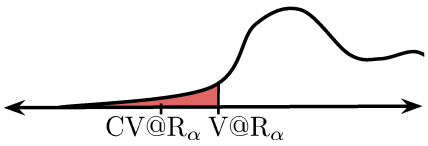


Figure 1: VaR_α measures the $(1-\alpha)$-quantile worst-case outcome in a distribution. CVaR_α measures the expectation given that we only consider values less than the VaR_α.

Conditional Value at Risk CVaR is a coherent risk measure [19] that is also commonly referred to as average value at risk, expected tail risk, or expected shortfall. For continuous atomless distributions, the CVaR is defined as

$$\mathrm{CVaR}_\alpha[X] = \mathbb{E}\left[X \mid X \leq \mathrm{VaR}_\alpha[X]\right]. \tag{1}$$

In addition to being coherent, CVaR is convex, and is a lower bound on VaR. CVaR is often preferable over VaR because it does not ignore the tail of the distribution and it is convex [50].

4 Balancing Risk and Return for Safe Imitation Learning

Let Π be the set of all randomized policies, and let $\mathcal{R}$ be the set of all reward functions. Given some function $\psi : \Pi \times \mathcal{R} \to \mathbb{R}$ representing any performance metric for a policy under the unknown

reward function $R \sim \mathbb{P}(R)$, we seek to find the policy that is the solution to the following problem:

$$\max_{\pi \in \Pi} \ \mathrm{CVaR}_\alpha[\psi(\pi, R)] \tag{2}$$

The obvious choice for the performance metric is $\psi(\pi, r) = \rho(\pi, r)$. We discuss other choices in Section 4.2. We now discuss how to solve for the policy that optimizes Equation (2). We build on the classic LP formulation of MDP planning, which optimizes the state occupancy distribution subject to the Bellman flow constraints [47]. Specifically, we make use of the one-to-one correspondence between randomized policies $\pi : \mathcal{S} \to \Delta^A$ (where A is the number of actions) and the state-action occupancy frequencies $\boldsymbol{u}_\pi$ [47]. This allows us to write $\max_\pi \rho(\pi, r)$ as the following linear program [47, 59]:

$$\max_{\boldsymbol{u} \in \mathbb{R}^{SA}} \left\{ \boldsymbol{r}^\mathsf{T} \boldsymbol{u} \ \mid \ \sum_{a \in \mathcal{A}} (\boldsymbol{I} - \gamma \cdot \boldsymbol{P}_a^\mathsf{T}) \boldsymbol{u}^a = \boldsymbol{p}_0, \boldsymbol{u} \geq \boldsymbol{0} \right\} . \tag{3}$$

We denote the posterior distribution over samples from $\mathbb{P}(R \,|\, D)$ as the vector $\boldsymbol{p}_R$, where each element of $\boldsymbol{p}_R$ represents the probability mass of one of the samples from the posterior distribution, e.g., $\boldsymbol{p}_R[i] = 1/N$ for N sampled reward functions $R_1, R_2, R_3, \ldots R_N$ obtained via MCMC [12, 48]. Because posterior distributions obtained via Bayesian IRL are usually discrete [12, 27, 48, 53], we cannot directly optimize for CVaR using the definition in (1) since this definition only works for atomless distributions (i.e. most continuous distributions). Instead, we use the following convex definition of CVaR [50] that works for any distribution (discrete or continuous):

$$\mathrm{CVaR}_\alpha[X] = \max_{\sigma \in \mathbb{R}} \ \left(\sigma - \frac{1}{1-\alpha} \mathbb{E}[(\sigma - X)_+] \right) , \tag{4}$$

where $(x)_+ = \max(0, x)$ and the optimal σ is equal to VaR_α for atomless distributions [50]. Although we focus on the CVaR measure, our approach readily extends to other convex risk measures, such as the entropic risk [56]. The only difference is that our linear programs turn to tractable convex optimizations.

Writing the convex definition of CVaR in terms of a the probability mass vector $\boldsymbol{p}_R \in \mathbb{R}^N$, results in the following definition of the CVaR of a policy π under the performance metric $\psi : \Pi \times \mathcal{R} \to \mathbb{R}$ and reward function random variable R:

$$\mathrm{CVaR}_\alpha[\psi(\pi, R)] = \max_{\sigma \in \mathbb{R}} \ \left(\sigma - \frac{1}{1-\alpha} \mathbb{E}\big[\big(\sigma - \psi(\pi, R)\big)\big]_+ \right) \tag{5}$$

$$= \max_{\sigma \in \mathbb{R}} \ \left(\sigma - \frac{1}{1-\alpha} \boldsymbol{p}_R^\mathsf{T} [\sigma \cdot \boldsymbol{1} - \boldsymbol{\psi}(\pi, R)]_+ \right) , \tag{6}$$

where the boldface $\boldsymbol{\psi}(\pi, R) = \big(\psi(\pi, R_1), \ldots, \psi(\pi, R_N)\big)^\mathsf{T}$ and $[\cdot]_+$ denotes the element-wise non-negative part of a vector: $[\boldsymbol{y}]_+ = \max\{\boldsymbol{y}, \boldsymbol{0}\}$. When the posterior distribution over R is continuous, Equation (6) represents the Sample Average Approximation (SAA) method applied to (5), which is used extensively in stochastic programming [56] with known finite-sample properties [9]. One of the main insights of this chapter is that, using the same approach as the linear program above, we can formulate (2) as the following linear program which can be solved in polynomial time:

$$\max_{\boldsymbol{u} \in \mathbb{R}^{SA}, \sigma \in \mathbb{R}} \left\{ \sigma - \frac{1}{1-\alpha} \boldsymbol{p}_R^\mathsf{T} \left[\sigma \cdot \boldsymbol{1} - \boldsymbol{\psi}(\pi, R)\right]_+ \ \mid \ \sum_{a \in \mathcal{A}} (\boldsymbol{I} - \gamma \cdot \boldsymbol{P}_a^\mathsf{T}) \boldsymbol{u}^a = \boldsymbol{p}_0, \boldsymbol{u} \geq \boldsymbol{0} \right\} . \tag{7}$$

Given the state-action occupancies $\boldsymbol{u}$ that maximize the above objective, the optimal policy can be recovered by appropriately normalizing these occupancies [47]. Thus, the optimal risk-averse IRL policy $\pi^\star$ can be constructed from an optimal $\boldsymbol{u}^\star$ solution to (7) as:

$$\pi^\star(s, a) = \frac{\boldsymbol{u}^\star(s, a)}{\sum_{a' \in \mathcal{A}} \boldsymbol{u}^\star(s, a')} . \tag{8}$$

4.1 Balancing Robustness and Expected Return

The above formulation in (7) finds a policy that has maximum CVaR. While this makes sense for highly risk-sensitive domains such as autonomous driving [53, 63] or medicine [5, 33], in other

domains such as a robot vacuuming office carpets, we may also be interested in efficiency and performance, rather than pure risk-aversion. Even in highly risky situations, completely ignoring expected return and optimizing only for low probability events can lead to nonsensical behaviors that are overly cautious, such as an autonomous car deciding to never merge onto a busy highway [41].

To tune the risk-sensitivity of the optimized policy, we seek to solve for the policy that optimally balances performance and epistemic risk over the reward function. We formalize our goal via the parameter $\lambda \in [0, 1]$ and seek the policy that is the maximizer of the following optimization problem:

$$\max_{\pi \in \Pi} \quad \lambda \cdot \mathbb{E}[\psi(\pi, R)] + (1 - \lambda) \cdot \mathrm{CVaR}_\alpha[\psi(\pi, R)] \; . \tag{9}$$

When $\lambda = 0$ we recover the fully robust policy, when $\lambda \in (0, 1)$ we obtain soft-robustness, and when $\lambda = 1$ we recover the risk-neutral Bayesian optimal policy [48]. We refer to the generalized problem in Equation (9) as *Bayesian Robust Optimization for Imitation Learning* or BROIL. Finally, by reformulating the optimization problem in Equation (7), we formulate BROIL as the following linear program:

$$\begin{aligned} \underset{\boldsymbol{u} \in \mathbb{R}^{SA},\, \sigma \in \mathbb{R}}{\text{maximize}} \quad & \lambda \cdot \boldsymbol{p}_R^\mathsf{T} \boldsymbol{\psi}(\pi_{\boldsymbol{u}}, R) + (1 - \lambda) \cdot \left(\sigma - \frac{1}{1 - \alpha} \boldsymbol{p}_R^\mathsf{T} \left[\sigma \cdot \mathbf{1} - \boldsymbol{\psi}(\pi_{\boldsymbol{u}}, R) \right]_+ \right) \\ \text{subject to} \quad & \sum_{a \in \mathcal{A}} \left(\boldsymbol{I} - \gamma \cdot \boldsymbol{P}_a^\mathsf{T} \right) \boldsymbol{u}^a = \boldsymbol{p}_0, \quad \boldsymbol{u} \geq \mathbf{0} \; , \end{aligned} \tag{10}$$

where we denote the stochastic policy that corresponds to a state-action occupancy vector $\boldsymbol{u}$ as $\pi_{\boldsymbol{u}}$.

4.2 Measures of Robustness

BROIL provides a general framework for optimizing policies that trade-off risk and return based on the specific choice of random variable $\psi(\pi, R)$, representing the desired measure of the safety or performance of a policy. We next describe two natural choices for defining $\psi(\pi, R)$.

Robust Objective If we seek a policy that is robust over the distribution $\mathbb{P}(R)$, we should optimize CVaR with respect to $\psi(\pi, R) = \rho(\pi, R)$, the expected return of the policy. Note that R is a random variable so $\rho(\pi, R)$ is also a random variable that depends on the posterior distribution over R and on π. In terms of the linear program (10) above we have $\boldsymbol{\psi}(\pi_{\boldsymbol{u}}, R) = \boldsymbol{R}^\mathsf{T} \boldsymbol{u}$, where $\boldsymbol{R}$ is a matrix of size $(S \cdot A) \times N$ where each column of $\boldsymbol{R}$ represents one sample of the vector over rewards for each state and action pair.

Robust Baseline Regret Objective If we have a baseline such as an expert policy or demonstrated trajectories we may want maximize CVaR with respect to $\psi(\pi, R) = \rho(\pi, R) - \rho(\pi_E, R)$. This form of BROIL seeks to maximize the margin between the performance of the policy and the performance of the demonstrator. Rather than seeking to match the risk of the demonstrator [35], the Baseline Regret form of BROIL baselines its performance with respect to the random variable $\rho(\pi_E, R)$, while still trying to minimize tail risk. In terms of the linear program (10) above we have $\boldsymbol{\psi}(\pi_{\boldsymbol{u}}, R) = \boldsymbol{R}^\mathsf{T}(\boldsymbol{u} - \boldsymbol{u}_E)$. In practice, we typically only have samples of expert behavior rather than a full policy. In this case, we compute the empirical expected feature counts using a set of demonstrated trajectories $D = \{\tau_1, \ldots, \tau_m\}$ to get $\hat{\boldsymbol{\mu}}_E = \frac{1}{|D|} \sum_{\tau \in D} \sum_{(s_t, a_t) \in \tau} \gamma^t \phi(s_t, a_t)$, where $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^k$ denotes the reward features. We then solve the above linear program (10) with with the performance metric $\boldsymbol{\psi}(\pi_{\boldsymbol{u}}, R) = \boldsymbol{R}^\mathsf{T} \boldsymbol{u} - \boldsymbol{W}^\mathsf{T} \hat{\boldsymbol{\mu}}_E$, where $\boldsymbol{W}$ is a matrix of size k-by-N where each column $\boldsymbol{w}_i \in \mathbb{R}^k, i = 1, \ldots, N$ is a feature weight vector corresponding to each linear reward function R_i sampled from the posterior such that $R_i = \boldsymbol{\Phi} \boldsymbol{w}_i$.

5 Experiments

In the next two sections we explore two case studies that highlight the performance and benefits of using BROIL for robust policy optimization. For the sake of interpretability, we keep the case studies simple; however, BROIL easily scales to much larger problems due to the efficiency of linear programming solvers. In the Appendix we empirically study the runtime of BIRL and demonstrate

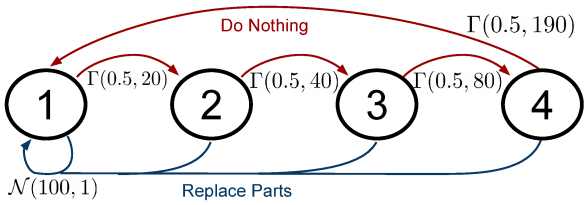


Figure 2: Machine Replacement MDP

that BROIL can efficiently solve problems involving thousands of states in only a few hundred seconds of compute time on a personal laptop.[2]

5.1 Zero-shot Robust Policy Optimization

We first consider the case where an agent wants to optimize a robust policy with respect to a prior over reward functions without access to expert demonstrations. This prior could come from historical data or from meta-learning on similar tasks [65].

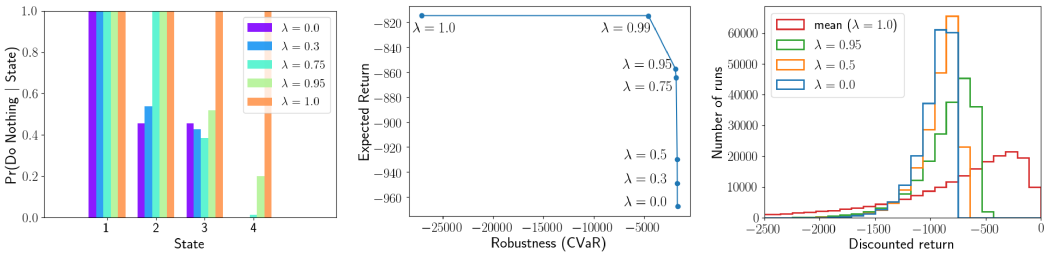
We consider the machine replacement problem, a common problem in the robust MDP literature [18]. In this problem, there is a factory with a large number of machines with parts that are expensive to replace. There is also a cost associated with letting a machine age without replacing parts as this may cause damage to the machine, but this cost is uncertain. We model this problem as the MDP shown in Figure 2 with 4 states that represent the normal aging process of the machine, two actions in each state (replace parts or do nothing), discount factor $\gamma = 0.95$, and uniform initial state distribution. The prior distribution over the cost of the Do Nothing action is modeled as a gamma distribution $\Gamma(x, \theta)$, resulting in low expected costs but increasingly large tails as the machine ages. The prior distribution over the cost of replacing a part is modeled using a normal distribution.

Because we have no demonstrations, we use the Robust Objective version of BROIL (Section 4.2). We sampled 2000 reward functions from the prior distributions over costs and computed the CVaR optimal policy with $\alpha = 0.99$ for different values of λ. Figure 3(a) shows the action probabilities of the optimal policy under different values of λ, where $\mathbb{P}(\text{Replace Parts}) = 1 - \mathbb{P}(\text{Do Nothing})$. Setting $\lambda = 1$ gives the optimal policy with respect to the mean reward under the reward posterior. This policy is risk-neutral and chooses to never repair the machine since the mean of the gamma distribution is $x \cdot \theta$, so in expectation it is optimal to do nothing. As λ decreases, the optimal policy hedges more against tail risk via a stochastic policy that sometimes repairs the machine. With $\lambda = 0$, we recover the robust optimal policy that only seeks to optimize CVaR. This policy is maximally risk-sensitive and chooses to probabilistically repair the machine in states 2 and 3 and always repair in state 4 to avoid the risk of doing nothing and incurring a possibly high cost. Figure 3(b) shows the efficient frontier of Pareto optimal solutions. BROIL achieves significant improvements in robustness by sacrificing a small amount of expected utility. Figure 3(c) shows that the BROIL policies with $\lambda < 1$ have much smaller tails than the policy that only optimizes with respect to the expected rewards.

5.2 Ambiguous Demonstrations

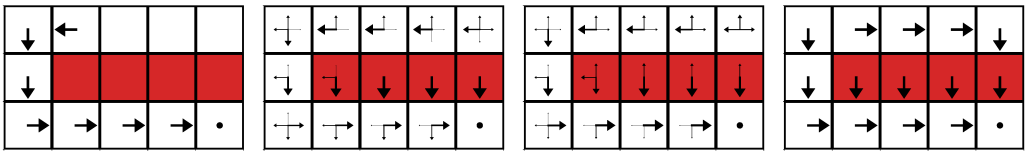
Next we consider the case where the agent has no prior knowledge about the reward function, but where demonstrations are available. In particular, we are interested in the case where demonstrations cover only part of the state-space, so even after observing a demonstration there is still high uncertainty over the true reward function. To clearly showcase the benefits of BROIL, we constructed the MDP shown in Figure 4 where there are two features (red and white) with unknown costs, a terminal state in the bottom right, and $\gamma = 0.95$. Actions are in the four cardinal directions with deterministic dynamics. The agent observes the demonstration shown in Figure 4(a) that demonstrates some preference for the white feature over the red feature and a preference for exiting the MDP. However, the demonstration does not provide sufficient information to know what to do in the top right states where demonstrator actions are unavailable. In particular, the agent does not know the true cost of the red cells and whether taking the shortest path from the top right states to the terminal state is

[2]Code to reproduce experiments is available at `https://github.com/dsbrown1331/broil`

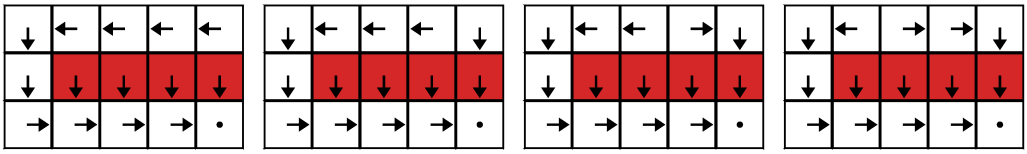


(a) Action Probabilities (b) Efficient Frontier (c) Return Distributions

Figure 3: Risk-sensitive ($\lambda \in [0, 1)$) and risk-neutral ($\lambda = 1$) policies for the machine replacement problem. Varying λ results in a family of solutions that trade-off conditional value at risk and return. The risk-neutral policy has heavy tails, while BROIL produces risk-sensitive policies that trade-off a small decrease in expected return for a large increase in robustness (CVaR).



(a) Ambiguous demo (b) MaxEnt IRL [67] (c) LPAL [59] (d) Robust ($\lambda = 0$)

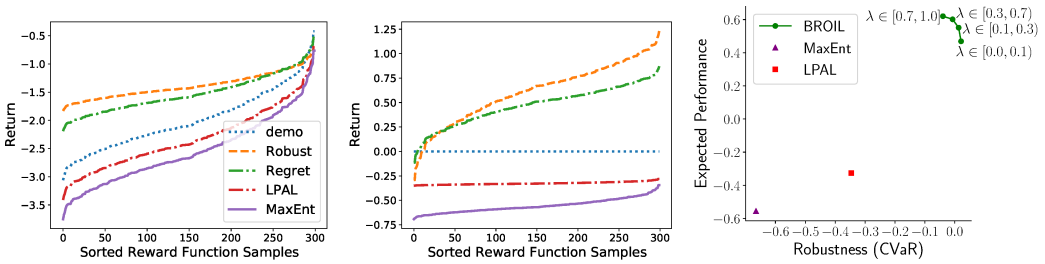


(e) Regret ($\lambda = 0$) (f) Regret ($\lambda = 0.1$) (g) Regret ($\lambda = 0.3$) (h) Robust/Regret ($\lambda = 1$)

Figure 4: When demonstrations BROIL results in a family of solutions that balance return and risk based on the value of λ. (a) Ambiguous demonstration that does not convey enough information to determine how undesireable the red states are. (b-c) MaxEnt IRL and LPAL results in stochastic policies where size of arrow reprents probability. (d) The robust policy with $\lambda = 0$ balances the goodness and badness of red and prefers taking a shortcut. (e-g) The regret policy avoids red for small λ. (h) The optimal policy for the mean reward ($\lambda = 1$) takes a short cut through red cells.

optimal. We demonstrate that BROIL results in much more sensible policies across a spectrum of risk-sensitivies, than other state-of-the-art approaches.

Given the single demonstration, we generated 2000 samples from the posterior $\mathbb{P}(R \mid D)$ using Bayesian IRL [48]. We compare against the risk-sensitive, maxmin algorithm, LPAL, proposed by Syed et al. [59] and the risk-neutral Maximum Entropy IRL algorithm [67]. Shown in Figure 4 are the optimal policies for MaxEnt IRL [67], LPAL [59], and BROIL using the robust and baseline regret formulations with $\alpha = 0.95$. We plotted the unique policies and a sample λ that results in each policy. Note that $\lambda = 1$ is equivalent to solving for the optimal policy for the mean reward Figure 4(h). The baseline regret formulation uses the expert feature counts to baseline risk and seeks to completely avoid the red feature for $\lambda = 0$. As λ increases, the baseline regret policy is more willing to take a shortcut to get to the terminal state in the bottom right corner. Conversely, the robust policy takes the shortcut through the far right red cell which balances the risk of the red feature with the knowledge that the white feature is likely to also have high cost. The reason the robust policy does not match the demonstration for one state in Figure 4(d) is that Bayesian IRL does not assume demonstrator optimality, only Boltzman rationality. We used a relatively small inverse temperature parameter ($\beta = 10$) resulting in reward function hypotheses that allow for occasional demonstrator errors. Using a large inverse temperature causes the robust policy to match all the demonstrator's actions (see the Appendix for more details regarding Bayesian IRL).



(a) Robustness (b) Performance over baseline (c) Efficient frontier

Figure 5: Sorted return distributions over the posterior for the BROIL Robust and Baseline Regret policies compared to the return distributions of the demonstration, MaxEnt IRL [67], LPAL [59]. The robust policy attempts to maximize worst-case performance over the posterior. The baseline regret also seeks to maximize worst-case performance but relative to the demonstration.

To better understand the differences between these approaches without committing to a particular ground-truth reward function, we examine each algorithm's performance across the posterior distribution $\mathbb{P}(R \mid D)$. Figure 5(a) shows $\psi(\pi, R) = \rho(\pi, R)$ sorted from smallest to largest when evaluated under each sample from the posterior. Figure 5(b) shows the results when $\psi(\pi, R) = \rho(\pi, R) - \rho(\pi_E, R)$. LPAL is similar to the baseline regret formulation of BROIL in that it seeks to optimize a policy that performs better than the demonstrator; however, unlike BROIL, LPAL uses a fully adversarial maxmin approach that penalizes the biggest deviation from the demonstrated feature counts [59]. This results in always avoiding red cells, but also trying to exactly match the feature counts of the demonstration. This feature count matching results in a highly stochastic policy that does not always terminate quickly. MaxEnt IRL is completely risk-neutral, but also seeks to explicitly match feature counts while maintaining maximum entropy over the policy actions. This results in a highly stochastic policy that sometimes takes shortcuts through the red cells, but also sometimes takes actions that move it away from the terminal state.

Figure 5 shows that both formulations of BROIL significantly outperform MaxEnt IRL and LPAL. The return distribution of the robust BROIL policy is flatter than the other policies as it attempts to find a policy that performs well in the 5% worst-case under all reward functions and needs to be robust to posterior samples that put high costs on white cells and only slightly higher costs on red. On the other hand the Baseline Regret formulation computes risk under the posterior with respect to the expected feature counts $\hat{\boldsymbol{\mu}}_E$ of the demonstrator. This makes reward function hypotheses that would lead to entering red states more risky since the demonstrator only visited white states. The regret formulation seeks to maximize the margin between the return of the baseline regret policy and the return of the demonstration over the posterior. Thus, the regret policy tracks the performance of the baseline more than the robust policy as shown in Figure 5(a). As shown in Figure 5(b), the regret formulation has better tail performance with respect to the posterior baseline regret. Figure 5(c) shows the efficient frontier for the baseline regret formulation and shows that BROIL dominates LPAL and MaxEnt IRL with respect to both expected return and robustness.

6 Conclusion and Future Work

We proposed Bayesian Robust Optimization for Imitation Learning (BROIL), a method for optimizing a policy to be robust to conditional value at risk under an unknown reward function. Our results show that BROIL has better overall performance than existing risk-sensitive maxmin [59] and risk-neutral [67] approaches to IRL. Our approach balances return and conditional value at risk to produce a family of robust solutions parameterized by the risk-aversion of the user. This work focuses on policy optimization and requires either a prior or posterior distribution over likely reward functions. However, obtaining a posterior via Bayesian IRL [48] typically involves repeatedly solving an MDP in the inner loop which makes it difficult to obtain posterior distributions in complex control tasks. Future work includes taking advantage of recent research on efficient non-linear Bayesian reward learning via Gaussian processes [10] and deep neural networks [13]. Future work also includes investigating natural extensions of our work to continuous state and action spaces such as optimizing the BROIL objective via policy gradient methods [55, 58] or approximate linear

programming [21, 43, 44], applying BROIL to more complex domains such as health care and robotics, and investigating extensions to deep Bayesian inverse reinforcement learning [13], meta inverse reinforcement learning [65], and inverse reward design [27].

Broader Impact

Algorithms that balance risk and return are have been common in financial applications for a long time, but are just starting to be applied to AI/ML systems. We believe this is a positive trend as many AI/ML applications have risk and return trade-offs that are not always adequately addressed. In this work we have proposed a principled approach optimizing control policies that balance expected return and epistemic risk under an uncertain reward functions. We see this work as an important step towards the general goal of robust autonomous systems that can interact safely with and assist humans in a wide variety of tasks and under a wide variety of preferences and risk tolerances. However, there are potential downsides to having risk and return trade-offs if these trade-offs are made incorrectly or interpreted incorrectly—despite using risk-sensitive metrics, financial systems still occasionally crash or fail. Our proposed algorithm, BROIL, does not guarantee safety, thus an autonomous system based on our approach will not be guaranteed to never make a mistake. Instead, BROIL optimizes a policy that is robust with respect to the agent's uncertainty over its learned representation of the demonstrator's reward function. Thus, the optimized policy may not always conform to a human's intuition about what safe or robust behavior should look like.

Acknowledgments and Disclosure of Funding

We would like to thank the reviewers for their detailed feedback that helped to improve the paper. This work has taken place in the Personal Autonomous Robotics Lab (PeARL) at the University of Texas at Austin and the Reinforcement Learning and Robustness Lab (RLsquared) at the University of New Hampshire. PeARL research is supported in part by the NSF (IIS-1724157, IIS-1638107,IIS-1617639, IIS-1749204) and ONR(N00014-18-2243). This research was also sponsored by the Army Research Office and was accomplished under Cooperative Agreement Number W911NF-19-2-0333. RLsquared research is supported in part by NSF Grants IIS-1717368 and IIS-1815275. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2004.

[2] Asrar Ahmed and Patrick Jaillet. Sampling Based Approaches for Minimizing Regret in Uncertain Markov Decision Processes (MDPs). *Journal of Artificial Intelligence Research (JAIR)*, 59:229–264, 2017.

[3] Asrar Ahmed, Padeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Regret based Robust Solutions for Uncertain Markov Decision Processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[4] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

[5] Hideki Asoh, Masanori Shiro1 Shotaro Akaho, Toshihiro Kamishima, Koiti Hasida, Eiji Aramaki, and Takahide Kohro. An application of inverse reinforcement learning to medical records of diabetes treatment. In *ECML-PKDD Workshop on Reinforcement Learning with Generalized Feedback*, 2013.

[6] Aharon Ben-Tal, Dmitris Bertsimas, and David B. Brown. A Soft Robust Model for Optimization Under Ambiguity. *Operations Research*, 58(4):1220–1234, 2010.

[7] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[8] James Berger. An overview of robust Bayesian analysis (with discussion). *Test*, 3(1):5–124, 1994.

[9] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. Robust sample average approximation. *Mathematical Programming*, 171(1-2):217–282, 2018.

[10] Erdem Biyik, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. Active preference-based gaussian process regression for reward learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[11] Daniel S. Brown, Yuchen Cui, and Scott Niekum. Risk-aware active inverse reinforcement learning. In *Conference on Robot Learning*, pages 362–372, 2018.

[12] Daniel S. Brown and Scott Niekum. Efficient probabilistic performance bounds for inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[13] Daniel S. Brown, Scott Niekum, Russell Coleman, and Ravi Srinivasan. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*. 2020.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. 2020.

[15] Jaedeug Choi and Kee-Eung Kim. MAP inference for Bayesian inverse reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1989–1997, 2011.

[16] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making : A CVaR optimization approach. In *Neural Information Processing Systems (NIPS)*, 2015.

[17] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Advances in Neural Information Processing Systems*, pages 1522–1530, 2015.

[18] Erick Delage and Shie Mannor. Percentile optimization for Markov decision processes with parameter uncertainty. *Operations Research*, 58(1):203–213, 2010.

[19] Freddy Delbaen. Coherent risk measures on general probability spaces. In *Advances in Finance and Stochastics*, pages 1–37. Springer, 2002.

[20] Esther Derman, Daniel Mankowitz, Timothy A Mann, and Shie Mannor. Soft-Robust Actor-Critic Policy-Gradient. In *Uncertainty in Artificial Intelligence (UAI)*, 2018.

[21] V. V. Desai, V. F. Farias, and C. C. Moallemi. Approximate dynamic programming via a smoothed linear program. *Operations Research*, 60(3):655–674, 2012.

[22] Hannes Eriksson and Christos Dimitrakakis. Epistemic risk-sensitive reinforcement learning. *arXiv preprint arXiv:1906.06273*, 2019.

[23] Hans Follmer and Alexander Schied. *Stochastic Finance: An Introduction in Discrete Time*. Walter de Gruyter, 3rd edition, 2011.

[24] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

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

[26] Vineet Goyal and Julien Grand-Clement. Robust Markov Decision Process: Beyond Rectangularity. Technical report, 2018.

[27] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6765–6774, 2017.

[28] Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. Partial policy iteration for L1-robust Markov decision processes. *preprint arXiv:2006.09484*, 2020.

[29] Jonathan Ho and Stefan Ermon. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 7461–7472, 2016.

[30] Jessie Huang, Fa Wu, Doina Precup, and Yang Cai. Learning safe policies with expert guidance. In *Advances in Neural Information Processing Systems*, pages 9105–9114, 2018.

[31] Dan A. Iancu and Nikolaos Trichakis. Pareto Efficiency in Robust Optimization. *Management Science*, 60(1):130–147, 2014.

[32] Philippe Jorion. *Value at Risk: A New Benchmark for Measuring Derivatives Risk*. Irwin Professional Publishers, 1996.

[33] John Kalantari, Heidi Nelson, and Nicholas Chia. The unreasonable effectiveness of inverse reinforcement learning in advancing cancer research. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

[34] Nathan Kallus and Angela Zhou. Confounding-Robust Policy Improvement. In *Neural Information Processing Systems (NIPS)*, 2018.

[35] Jonathan Lacotte, Mohammad Ghavamzadeh, Yinlam Chow, and Marco Pavone. Risk-sensitive generative adversarial imitation learning. In *Artificial Intelligence and Statistics (AISTATS)*, 2019.

[36] Romain Laroche, Paul Trichelair, Rémi Tachet des Combes, and Remi Tachet. Safe Policy Improvement with Baseline Bootstrapping. In *International Conference of Machine Learning (ICML)*, 2019.

[37] Anirudha Majumdar, Sumeet Singh, Ajay Mandlekar, and Marco Pavone. Risk-sensitive inverse reinforcement learning via coherent risk models. In *Robotics: Science and Systems*, 2017.

[38] Shie Mannor, Ofir Mebel, and Huan Xu. Robust MDPs with k-rectangular uncertainty. *Mathematics of Operations Research*, 41(4):1484–1509, 2016.

[39] Harry M Markowitz and G Peter Todd. *Mean-variance analysis in portfolio choice and capital markets*. John Wiley & Sons, 2000.

[40] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000.

[41] Thwarted on the On-ramp: Waymo Driverless Car Doesn't Feel the Urge to Merge. https://www.thetruthaboutcars.com/2018/05/thwarted-ramp-waymo-driverless-car-doesnt-feel-urge-merge/. (accessed: 05.19.2020).

[42] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

[43] Jason Pazis and Ronald Parr. Non-parametric Approximate Linear Programming for MDPs. In *Conference on Artificial Intelligence (AAAI)*, 2011.

[44] Marek Petrik. *Optimization-based Approximate Dynamic Programming*. PhD thesis, University of Massachusetts Amherst, 2010.

[45] Marek Petrik, Mohammad Ghavamzadeh, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems*, pages 2298–2306, 2016.

[46] Marek Petrik and Dharmashankar Subramanian. An approximate solution method for large risk-averse Markov decision processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2012.

[47] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. 2005.

[48] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 2586–2591, 2007.

[49] Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 444–451, 2009.

[50] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[51] Reazul Hasan Russell and Marek Petrik. Beyond Confidence Regions: Tight Bayesian Ambiguity Sets for Robust MDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[52] Stuart Russell and Peter Norvig. Artificial intelligence: A modern approach. 2002.

[53] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

[54] Anirban Santara, Abhishek Naik, Balaraman Ravindran, Dipankar Das, Dheevatsa Mudigere, Sasikanth Avancha, and Bharat Kaul. RAIL : Risk-Averse Imitation Learning Extended Abstract. (D):2062–2063, 2018.

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

[56] A. Shapiro, D. Dentcheva, and A. Ruszczynski. *Lectures on stochastic programming: Modeling and theory*. 2014.

[57] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

[58] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1063, 2000.

[59] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. pages 1032–1039, 2008.

[60] Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. In *AAAI Conference on Artificial Intelligence*, 2015.

[61] Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. Worst cases policy gradients. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, Proceedings of Machine Learning Research, pages 1078–1093. PMLR, 2020.

[62] Andrea Tirinzoni, Xiangli Chen, Marek Petrik, and Brian D Ziebart. Policy-Conditioned Uncertainty Sets for Robust Markov Decision Processes. In *Neural Information Processing Systems (NIPS)*, 2018.

[63] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017.

[64] Huan Xu and Shie Mannor. Parametric regret in uncertain Markov decision processes. In *IEEE Conference on Decision and Control (CDC)*, pages 3606–3613, 2009.

[65] Kelvin Xu, Ellis Ratner, Anca Dragan, Sergey Levine, and Chelsea Finn. Learning a prior over intent via meta-inverse reinforcement learning. *International Conference on Machine Learning*, 2019.

[66] Jiangchuan Zheng, Siyuan Liu, and Lionel M Ni. Robust Bayesian inverse reinforcement learning with sparse behavior noise. In *AAAI Conference on Artificial Intelligence*, 2014.

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

A Code

Code to reproduce all experiments is available at `https://github.com/dsbrown1331/broil`.

B Linear Programming Details

The soft-robust BROIL objective is:

$$\begin{aligned} \underset{\boldsymbol{u}\in\mathbb{R}^{SA},\,\sigma\in\mathbb{R}}{\text{maximize}} \quad & \lambda\cdot\boldsymbol{p}^{\mathsf{T}}\boldsymbol{\psi}(\pi_{\boldsymbol{u}},R)+(1-\lambda)\cdot\left(\sigma-\frac{1}{1-\alpha}\boldsymbol{p}^{\mathsf{T}}\left[\sigma\cdot\mathbf{1}-\boldsymbol{\psi}(\pi_{\boldsymbol{u}},R)\right]_{+}\right) \\ \text{subject to} \quad & \sum_{a\in\mathcal{A}}\left(\boldsymbol{I}-\gamma\cdot\boldsymbol{P}_a^{\mathsf{T}}\right)\boldsymbol{u}^a=\boldsymbol{p}_0, \quad \boldsymbol{u}\geq\mathbf{0}\,. \end{aligned}$$

When using the robust performance metric described in Section 4.2, we have $\boldsymbol{\psi}(\pi_{\boldsymbol{u}},R)=\boldsymbol{R}^{\mathsf{T}}\boldsymbol{u}$, where $\boldsymbol{R}$ is a matrix of size $(S\cdot A)\times N$ where each column of $\boldsymbol{R}$ represents one sample of the vector over rewards for each state and action pair. This results in the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{u}\in\mathbb{R}^{SA},\,\sigma\in\mathbb{R}}{\text{maximize}} \quad & \lambda\cdot(\boldsymbol{R}\boldsymbol{p})^{\mathsf{T}}\boldsymbol{u}+(1-\lambda)\cdot\left(\sigma-\frac{1}{1-\alpha}\boldsymbol{p}^{\mathsf{T}}\left[\sigma\cdot\mathbf{1}-\boldsymbol{R}^{\mathsf{T}}\boldsymbol{u}\right]_{+}\right) \\ \text{subject to} \quad & \sum_{a\in\mathcal{A}}\left(\boldsymbol{I}-\gamma\cdot\boldsymbol{P}_a^{\mathsf{T}}\right)\boldsymbol{u}^a=\boldsymbol{p}_0, \quad \boldsymbol{u}\geq\mathbf{0}\,. \end{aligned}$$

where $\boldsymbol{R}\boldsymbol{p}$ is the mean reward under the posterior distribution.

This can be written as a linear program in standard form as:

$$\begin{aligned} -\underset{\boldsymbol{u}\in\mathbb{R}^{SA},\,\boldsymbol{z}\in\mathbb{R}^{N},\,\sigma\in\mathbb{R}}{\text{minimize}} \quad & -\lambda\cdot\boldsymbol{p}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{u}-(1-\lambda)\cdot(\sigma+\frac{1}{1-\alpha}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{z}) \\ \text{subject to} \quad & \sigma\cdot\mathbf{1}-\boldsymbol{R}^{\mathsf{T}}\boldsymbol{u}-\boldsymbol{z}\leq 0, \\ & \left[(\boldsymbol{I}-\gamma\boldsymbol{P}_{a_1}^{\mathsf{T}}),\ldots,(\boldsymbol{I}-\gamma\boldsymbol{P}_{a_m}^{\mathsf{T}})\right]\begin{bmatrix}\boldsymbol{u}^{a_1}\\ \vdots\\ \boldsymbol{u}^{a_n}\end{bmatrix}=\boldsymbol{p}_0, \\ & \boldsymbol{u}\geq\mathbf{0},\ \boldsymbol{z}\geq\mathbf{0}\,. \end{aligned}$$

We solve the above linear program to obtain the results presented in Section 5.1.

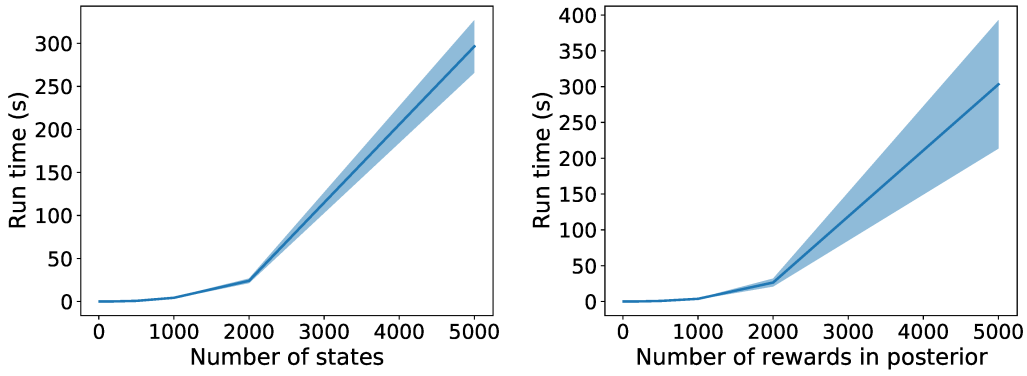
When using the baseline regret performance metric, $\boldsymbol{\psi}(\pi_{\boldsymbol{u}},R)=\boldsymbol{R}^{\mathsf{T}}(\boldsymbol{u}-\boldsymbol{u}_E)$, we have the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{u}\in\mathbb{R}^{SA},\,\sigma\in\mathbb{R}}{\text{maximize}} \quad & \lambda\cdot(\boldsymbol{R}\boldsymbol{p})^{\mathsf{T}}(\boldsymbol{u}-\boldsymbol{u}_E)+(1-\lambda)\cdot\left(\sigma-\frac{1}{1-\alpha}\boldsymbol{p}^{\mathsf{T}}\left[\sigma\cdot\mathbf{1}-\boldsymbol{R}^{\mathsf{T}}(\boldsymbol{u}-\boldsymbol{u}_E)\right]_{+}\right) \\ \text{subject to} \quad & \sum_{a\in\mathcal{A}}\left(\boldsymbol{I}-\gamma\cdot\boldsymbol{P}_a^{\mathsf{T}}\right)\boldsymbol{u}^a=\boldsymbol{p}_0, \quad \boldsymbol{u}\geq\mathbf{0}\,, \end{aligned}$$

This can be written as a linear program in standard form as follows:

$$\begin{aligned} -\underset{\boldsymbol{u}\in\mathbb{R}^{SA},\,\boldsymbol{z}\in\mathbb{R}^{N},\,\sigma\in\mathbb{R}}{\text{minimize}} \quad & -\lambda\cdot\boldsymbol{p}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}(\boldsymbol{u}-\boldsymbol{u}_E)-(1-\lambda)\cdot(\sigma+\frac{1}{1-\alpha}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{z}) \\ \text{subject to} \quad & \sigma\cdot\mathbf{1}-\boldsymbol{R}^{\mathsf{T}}\boldsymbol{u}-\boldsymbol{z}\leq-\boldsymbol{R}^{\mathsf{T}}\boldsymbol{u}_E \\ & \left[(\boldsymbol{I}-\gamma\boldsymbol{P}_{a_1}^{\mathsf{T}}),\ldots,(\boldsymbol{I}-\gamma\boldsymbol{P}_{a_m}^{\mathsf{T}})\right]\begin{bmatrix}\boldsymbol{u}^{a_1}\\ \vdots\\ \boldsymbol{u}^{a_n}\end{bmatrix}=\boldsymbol{p}_0 \\ & \boldsymbol{u}\geq\mathbf{0},\ \boldsymbol{z}\geq\mathbf{0}\,. \end{aligned}$$

Typically, we only have access to a handful of demonstrations and do not have direct access to the state-occupancies of the demonstrator and cannot accurately estimate them. If we assume the reward



(a) (b)

Figure 6: LP runtime as number of states and number of reward function hypothesis are increased for the machine replacement problem. Results are averaged over 20 trials and error bars show plus or minus one standard deviation. (a) Runtime as the number of states is increased and number of reward function hypotheses is fixed at 200. (b) Runtime as the number of reward function hypotheses is increased and the number of states is fixed at 100.

function is a linear combination of features, it is often the case that the number of features k is much less than the total number of state-action pairs. Thus, it is typically much more practical and computationally accurate to use an empirical estimate of the expert's expected feature counts and rewrite the baseline regret as $\psi(\pi_{\boldsymbol{u}}, R) = \boldsymbol{R}^\mathsf{T}\boldsymbol{u} - \boldsymbol{W}^\mathsf{T}\hat{\boldsymbol{\mu}}_E$, where $\boldsymbol{W}$ is a matrix of size k-by-N where each column is a feature weight vector $\boldsymbol{w} \in \mathbb{R}^k$ corresponding to each linear reward function weight vector sampled from the posterior as described in Section 4.2. This results in the following linear program which we use for our experiments in Section 5.2:

$$
\begin{aligned}
-\underset{\boldsymbol{u}\in\mathbb{R}^{SA},\, \boldsymbol{z}\in\mathbb{R}^{N},\, \sigma\in\mathbb{R}}{\text{minimize}} \quad & -\lambda \cdot \boldsymbol{p}^\mathsf{T}(\boldsymbol{R}^\mathsf{T}\boldsymbol{u} - \boldsymbol{W}^\mathsf{T}\hat{\boldsymbol{\mu}}_E) - (1-\lambda)\cdot(\sigma + \frac{1}{1-\alpha}\boldsymbol{p}^\mathsf{T}\boldsymbol{z}) \\
\text{subject to} \quad & \sigma \cdot \mathbf{1} - \boldsymbol{R}^\mathsf{T}\boldsymbol{u} - \boldsymbol{z} \leq -\boldsymbol{W}^\mathsf{T}\hat{\boldsymbol{\mu}}_E \\
& \left[(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1}^\mathsf{T}), \ldots, (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_m}^\mathsf{T})\right] \begin{bmatrix} \boldsymbol{u}^{a_1} \\ \vdots \\ \boldsymbol{u}^{a_n} \end{bmatrix} = \boldsymbol{p}_0 \\
& \boldsymbol{u} \geq \mathbf{0},\ \boldsymbol{z} \geq \mathbf{0}\,.
\end{aligned}
$$

We use Scipy's linear programming software (v 1.4.1) when solving the above linear programs in the experiments in the paper.[3] Note also that the term $-\lambda \boldsymbol{p}^\mathsf{T}\boldsymbol{R}^\mathsf{T}\boldsymbol{u}_E$ or $-\lambda \boldsymbol{p}^\mathsf{T}\boldsymbol{W}^\mathsf{T}\hat{\boldsymbol{\mu}}_E$ in the baseline regret linear program objectives above can be dropped since they are just constants that do not affect the resulting optimal policies.

C Runtime and Scalability

In the main paper we focused on simple case studies that are easily interpretable; however, our method readily scales to much larger problems. In Figure 6 we show that we can easily solve instances with thousands of states and thousands of reward functions in the posterior. To further test the runtime of BROIL, we optimized a robust policy for a 60-by-60 gridworld (3,600 states) which took an average time of 119.34 seconds to solve the BROIL linear program given a reward function distribution. For comparison, a CVaR optimization approach for MDPs with no uncertainty over the reward function takes 2 hours for a similar-sized gridworld (see [17] Section 5, last paragraph). All experiments were run using Scipy's standard linear programming solver on a Dell Inspiron 5577 laptop with an Intel i7-7700 processor.

[3] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html`

D Bayesian IRL Details

When learning a posterior from demonstrations we use Bayesian IRL [48]. Bayesian IRL has the following likelihood function: Bayesian IRL assumes access to an MDP without a reward function, denoted MDP\R and a set of demonstrations, $D = \{(s_1, a_1), \ldots, (s_m, a_m)\}$, consisting of state-action pairs. Bayesian IRL (BIRL) [48] seeks to estimate the posterior over reward functions given demonstrations, $\mathbb{P}(R \mid D) \propto \mathbb{P}(D \mid R) \cdot \mathbb{P}(R)$. BIRL makes the assumption that the demonstrator is Boltzmann rational and follows a soft-max policy, resulting in the likelihood function

$$\mathbb{P}(D \mid R) = \prod_{(s,a)\in D} \mathbb{P}\big((s,a) \mid R\big) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}} \tag{11}$$

where $Q^*_R(s,a)$ is the optimal Q-value function for reward R, and β is a parameter representing the confidence in the demonstrator's optimality. Given a reward function R, the Q-value of a state-action pair (s,a) is defined as $Q^\pi_R(s,a) = R(s) + \gamma \sum_{s'\in S} T(s,a,s') V^\pi_R(s')$. We denote $Q^*_R(s,a) = \max_{\pi\in\Pi} Q^\pi_R(s,a)$. Equation 11 gives greater likelihood to rewards for which the actions taken by the expert have higher Q-values than the alternative actions.

Bayesian IRL uses Markov chain Monte Carlo (MCMC) sampling to sample from the posterior $\mathbb{P}(R \mid D)$. Feature weights are sampled according to a proposal distribution, and for each sample the MDP is solved to obtain the sample's likelihood and determine the transition probabilities within the Markov chain. We use $\beta = 10$ for all of our experiments. We sample reward function weights for MCMC by using a Gaussian proposal distribution centered arounnd the previous sample, with standard deviation of 0.2. We use a burn-in period of 500 samples and skip every 5th sample after that to reduce auto-correlation. We experimented with range of values for β and found very similar results. The step size was tuned to result in an accept ratio close to 0.4. Because scaling a reward function does not affect the optimal policy, we following prior work [1, 12, 59] and assume that the reward function is scaled. We project each sampled reward function weight proposal to the L_2-norm ball to ensure that $\|w\|_2 = 1$.

E Maximum Entropy IRL Detais

We compare against Maximum Entropy IRL [67]. We use the implementation presented by Ziebart et al. [67], but to make it more comparable to Bayesian IRL we also add a Boltzmann parameter β to the likelihood such that

$$\mathbb{P}(\xi) \propto \exp(\beta R(\xi)). \tag{12}$$

where ξ is a trajectory and $R(\xi)$ is the cumulative return of a trajectory. We use $\beta = 10$ to match our implementation choice for Bayesian IRL. We used a learning rate of 0.01. We perform projected gradient descent by projecting to the L_2-norm ball such that $\|w\|_2 = 1$. We use a horizon equal to the number of states in the MDP. We stop gradient ascent on the likelihood function once it has converged. We detect convergence by measuring the difference in the L_2-norm of the updated and prior weights and check if that is within a precision value of 0.00001. If so, then we stop gradient ascent. We experimented with several different values for each of these hyperparameters and found these to provide best performance.

F LPAL Details

Linear Programming Apprenticeship Learning (LPAL) [59] has a robust form that is similar to ours but makes several critical and limiting assumptions: (1) they assume that the reward weights are strictly positive, this means they assume that the feature vector $\phi(s)$ explicitly encodes whether a feature is good or bad by its sign. (2) They assume very accurate estimation of the expert's expected feature counts $\hat{\mu}_E$. This requires an extremely large number of demonstrations (c.f. [12]). (3) Finally, they assume a worst-case adversarial reward function that penalizes whatever the learner does that is most different from the demonstrator, even if this reward function completely contradicts the demonstrations, i.e., it does not take into account the likelihood of reward functions.

To compare BROIL against a state-of-the-art robust IRL approach, we implemented Linear Programming Apprenticeship Learning [59]. The original paper assumes that the signs of the feature weights

determine whether a feature is good or bad and that the feature weights $\boldsymbol{w}$ lie on the probability simplex. In our work we do not assume prior knowledge about which features are good or bad (we seek to infer this from demonstrations). Thus, we implemented LPAL in a way that allows it to work with any features and feature weights that can be both positive and negative. We simply assume that $\|\boldsymbol{w}\|_1 \leq 1$.

In the paper we compare against the solution to the following derivation of the LPAL algorithm which does not assume the weights are non-negative, thus removing the need to know beforehand which features are good or bad. The LPAL formulation we use is as follows:

$$
\begin{aligned}
-\min_{B\in\mathbb{R},\, \boldsymbol{u}\in\mathbb{R}^{S\cdot A}} \quad & B && (13)\\
\text{s.t.} \quad & B\cdot\mathbf{1} - \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} \leq -\hat{\boldsymbol{\mu}}_E, && (14)\\
& -B\cdot\mathbf{1} + \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} \leq \hat{\boldsymbol{\mu}}_E, && (15)\\
& \left[(\boldsymbol{I}-\gamma\boldsymbol{P}_{a_1}^{\mathsf{T}}),\ldots,(\boldsymbol{I}-\gamma\boldsymbol{P}_{a_m}^{\mathsf{T}})\right]\begin{bmatrix}\boldsymbol{u}_{a_1}\\ \vdots \\ \boldsymbol{u}_{a_n}\end{bmatrix} = \boldsymbol{p}_0, && (16)\\
& \boldsymbol{u}\geq\mathbf{0}, && (17)\\
& B\in\mathbb{R}, && (18)
\end{aligned}
$$

where $\boldsymbol{\Phi}\in\mathbb{R}^{S\cdot A\times k}$ is the linear feature matrix with rows as states and columns as features and $\boldsymbol{w}\in\mathbb{R}^k$.

We now derive the above formulation of the maxmin objective for LPAL. The basic LPAL objective is

$$\max_{\boldsymbol{u}\in\mathcal{U}} \min_{\boldsymbol{w}\geq\mathbf{0}, \|\boldsymbol{w}\|_1\leq 1} (\boldsymbol{u}^{\mathsf{T}}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{u}_E^{\mathsf{T}}\boldsymbol{\Phi}\boldsymbol{w}), \tag{19}$$

where $\mathcal{U}$ is the set of all feasible state-action occupancies.

If we want to get rid of the requirement for positive weights then we have

$$\max_{\boldsymbol{u}\in\mathcal{U}} \min_{\|\boldsymbol{w}\|_1\leq 1} (\boldsymbol{u}^{\mathsf{T}}\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{u}_E^{\mathsf{T}}\boldsymbol{\Phi}\boldsymbol{w}) \tag{20}$$

The inner minimization can be changed into a maximization as follows:

$$\max_{\boldsymbol{u}\in\mathcal{U}} -\max_{\|\boldsymbol{w}\|_1\leq 1} (-\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} + \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u}_E)^{\mathsf{T}}\boldsymbol{w} \tag{21}$$

Next, we use the fact that the L_∞ norm and L_1 norm are dual to each other and that $\|\boldsymbol{z}\| = \|-\boldsymbol{z}\|$ to get the following optimization problem:

$$\max_{\boldsymbol{u}\in\mathcal{U}} -\|\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} - \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u}_E\|_\infty \tag{22}$$

We change the maximization to a minimization by changing signs:

$$-\min_{\boldsymbol{u}\in\mathcal{U}} \|\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} - \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u}_E\|_\infty \tag{23}$$

Using a standard linear programming reformulation we can write the above objective as follows:

$$-\min_{\boldsymbol{u}\in\mathcal{U},\, B\in\mathbb{R}} \left\{B \,|\, B\cdot\mathbf{1} \geq \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} - \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u}_E,\ -B\cdot\mathbf{1} \geq -\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} + \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u}_E\right\}. \tag{24}$$

Rather than assuming access to the state-action occupancies of the demonstrator, we will typically use a finite number of demonstrations to come up with an empirical estimate of the demonstrator's expected feature counts $\boldsymbol{\mu}_E = \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u}_E$. Given a set of demonstrated trajectories $D = \{\tau_1,\ldots,\tau_m\}$ we compute $\hat{\boldsymbol{\mu}}_E = \frac{1}{|D|}\sum_{\tau\in D}\sum_{(s_t,a_t)\in\tau}\gamma^t\phi(s_t,a_t)$, where $\phi : \mathcal{S}\times\mathcal{A}\to\mathbb{R}^k$ denotes the state-action reward features such that $r(s,a) = \boldsymbol{w}^{\mathsf{T}}\phi(s,a)$. This gives us the following linear program:

$$-\min_{\boldsymbol{u}\in\mathcal{U}, B\in\mathbb{R}} \left\{B \,|\, B\cdot\mathbf{1} \geq \boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} - \hat{\boldsymbol{\mu}}_E,\ -B\cdot\mathbf{1} \geq -\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{u} + \hat{\boldsymbol{\mu}}_E\right\}. \tag{25}$$