Robust Imitation Learning for Risk-Aware Behavior and Sim2Real Transfer

Zaynah Javed

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-48
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-48.html

May 10, 2022

Robust Imitation Learning for Risk-Aware Behavior and Sim2Real Transfer

by

Zaynah Badr Javed

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Anca Dragan

Spring 2022

Robust Imitation Learning for Risk-Aware Behavior and Sim2Real Transfer

by Zaynah Badr Javed

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Ken Goldberg
Research Advisor

10 May 2022

(Date)

* * * * * * *

Professor Anca Dragan
Second Reader

5/9/2022

(Date)

Robust Imitation Learning for Risk-Aware Behavior and Sim2Real Transfer

Abstract

Robust Imitation Learning for Risk-Aware Behavior and Sim2Real Transfer

by

Zaynah Badr Javed

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ken Goldberg, Chair

Learning from demonstrations circumvents the difficult and error-prone task of manually specifying a reward function. However, there are many issues that can arise. In some cases, not enough demonstration data exists. This can lead to ambiguous, imperfect demonstrations where the data gives rise to uncertainty over the true goal. There can be many different reward functions that explain this data, giving uncertainty over the true reward function that should be learned from the data. Most policy optimization approaches handle this uncertainty by optimizing for expected performance, but 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. Another issue that may arise with demonstrations is sim2real transfer, where demonstrations and training may be done via simulation, but the robot exists in the real world. Sim2Real transfer has emerged as a successful method to train robotic control policies for a wide variety of tasks, however it is often challenging to determine when policies trained in simulation are ready to be transferred to the physical world. Deploying policies which have been trained with very little simulation data can result in unreliable behaviors on real world hardware. On the other hand, excessive training in simulation can cause policies to overfit to the dynamics and visual appearance of the simulator. We study strategies to automatically determine when imitation learning policies trained in simulation can be reliably transferred to a physical robot. We study these ideas in the context of a robotic fabric manipulation task, in which successful sim2real transfer is challenging due to the difficulties of precisely modeling fabric.

To my grandparents.

Contents

List of Figures

List of Tables

Acknowledgments

Doing research for the last two years has been a transformative and incredible experience. I am forever grateful to my advisor Professor Ken Goldberg for taking me in at a time when he couldn't even meet me in person. He taught me how to challenge existing work, think ahead in research, and present with pride. I will never understand how he advises so many students, yet makes each one feel special.

I am beyond lucky to have been mentored by Daniel Brown. Working with Daniel has excited me about research in ways that words cannot describe. He has an incredible power to do top tier research and give the best support to newcomers. I know he will make a brilliant professor and can't wait to see what he does. I am also fortunate to have been guided by Ashwin Balakrishna. He genuinely seems to be able to do everything.

I have also been lucky to work closely with Satvik Sharma, Jerry Zhu, Ellen Novoseller, Vainavi Viswanath, Rishi Parikh, Albert Wilcox, and Ryan Hoque. They are all spectacular researchers and even better company. I also thank all members of the AUTOLab for being such an amazing intellectual community; I am constantly learning and improving because of all of you. There's never a dull moment in the lab.

I also would like to thank Professor Anca Dragan for being my second reader on this thesis and for providing insightful feedback on the original PG-BROIL paper.

I also want to thank Professor Ani Adhikari for being one of my favorite teachers of all time, and for giving me the invaluable opportunity to teach for Stat 140.

I am so lucky to have a close friend, Eric, doing this program with me. Thanks for being an amazing friend to go through this whole program with. As if carrying me for undergrad wasn't enough. I also want to thank all my friends I've made at Berkeley for filling the past 5 years on this campus with the best memories. Amy, Ashwin, Roop, Rania, Adeel, and Talha are just a few! I also want to especially thank my boyfriend Sajal for always supporting and believing in me.

I want to thank my family for all of their unconditional love and support. My parents for always being there and supportive of anything I choose to do. My sister, Nabiha, for always bringing fun into my life. My grandparents for always being in support of higher education. My aunts, uncles, and cousins for being second homes.

My largest takeaway from this past year isn't academic at all. It may surprise you (and concern my advisor) that during this past school year, school has been the least of my priorities. All the painful losses, difficult moments, and huge life changes over these last few months have taught me that life is too short to let it pass by. There is more to life than just your academic achievements (although those are significant too!). Live a life you want to remember, and a life that people will remember you by.

Chapter 1

Introduction

Imitation learning has seen wide success in real world applications [8, 71, 61, 28]. However the robustness of imitation learning is still challenged. Transferring learned policies to the real world proves to be difficult, since simulators are never perfect. As a result, learning algorithms must overcome a domain shift between the dynamics and visual appearance of simulations and the physical world. There has been a large body of prior work addressing different methods to enable robust domain adaptation of policies trained in simulation to physical experiments [87, 38, 73, 102, 46, 32]. However, less attention has been given to the critical challenge of determining when a policy trained in simulation is ready for physical deployment. While deploying imitation learning policies trained with a small amount of simulation data can lead to low performance on the physical system, policies that are trained excessively in simulation risk overfitting to artifacts in the simulator. This motivates principled methods to determine when policies trained in simulation are likely to be ready for physical deployment.

Determining when a policy is ready for physical deployment requires reasoning about its expected performance in physical trials when the policy has only been trained in simulation. One common way to achieve this is to periodically evaluate learned policies in the physical world during simulated training to determine when sufficient performance has been achieved [21, 43, 52]. However, this poses a number of practical challenges, as performing physical rollouts during training costs significant time and engineering effort. This motivates using policy rollouts only from simulation to determine when to deploy policies on a physical robot. While this setting is more practical, it is also significantly more difficult, since it is challenging to gain information about expected performance in physical trials at deployment time without access to the physical system prior to deployment. A key insight is that the problem of determining when to transfer robotic control policies from simulation to reality bears

a number of similarities to the problem of early-stopping, a well-known technique for reducing overfitting in machine learning [62, 50, 99]. For instance, Prechelt et al. 1999 [62] investigates 14 different early stopping criteria with cross validation, which assesses generalization by measuring performance on a validation dataset unseen by the learner. Meanwhile, other approaches do not rely on the existence of a validation set: Mahsereci et al. 2017 [50] derive an early stopping rule based on local gradient statistics and Yao et al. 2007 [99] propose a rule based on the bias-variance tradeoff. While early stopping is well-studied, however, few have explored applying the technique to sim2real transfer.

We perform a detailed empirical study of which metrics are predictive of robust policy transfer from simulation to the physical world and evaluate the proposed method for transferring fabric smoothing policies from simulation to deployment on a physical robot.

Reward function ambiguity is also a key problem in imitation learning [37, 58], 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 [60, 94] or take a maximum likelihood [14, 11] or maximum entropy approach to learning a reward function [103, 25], 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 [29, 12]. Bayesian inverse reinforcement learning (IRL) methods [64] seek a posterior distribution over likely reward functions given demonstrations, but often perform policy optimization using the expected reward function or MAP reward function [64, 14, 66, 12]. 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 [27, 84, 86, 19, 48, 91]. 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 [82, 67, 29, 36]. However, these approaches are limited to tabular domains, and maxmin approaches have been shown to sometimes lead to incorrect and overly pessimistic policy evaluations [13]. As an alternative to maxmin approaches, Bayesian Robust Optimization for Imitation Learning (BROIL), an approach that seeks to balance risk-aversion (in terms of Conditional Value at Risk [68]) and expected performance, was proposed. 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.

For this, 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 [10], we present an approach which reasons simultaneously about risk-aversion (in terms of Conditional Value at Risk [68]) 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 [78] or PPO [77] 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. In addition, we also derive an MPC formulation (MPC-BROIL) for model-based policy optimization.

We leverage prior work on Bayesian reward inference [12] to infer a posterior distribution over reward functions from human preferences over demonstrated trajectories. While other approaches which do not reason about reward uncertainty can sometimes overfit to a single reward function hypothesis, we demonstrate that optimizing the BROIL objective results in policies that hedge against multiple reward function hypotheses, leading to more robust performance. In particular, when there is high reward function ambiguity due to limited human feedback, we find that our approach results in significant performance improvements over other state-of-the-art imitation learning methods.

To evaluate PG-BROIL and MPC-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 our algorithms are able to optimize policies that effectively trade-off between expected and worst-case performance. Then, we leverage recent advances in efficient Bayesian reward inference [12] to infer a posterior over reward functions from preferences over demonstrated trajectories, and show that our algorithms perform best over these distributions as opposed to other algorithms (which may only consider the mean or MAP reward). While other approaches which do not reason about reward uncertainty overfit to a single reward function hypothesis, PG-BROIL and MPC-BROIL optimize a policy that hedges against multiple reward function hypotheses. When there is high reward function ambiguity due to limited demonstrations, we find that our algorithms result in significant performance improvements over other state-of-the-art imitation learning methods.

Chapter 2

Related Work

2.1 Reinforcement Learning

There has been significant recent interest in safe and robust reinforcement learning [27]; 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 [30, 80, 83, 85], establish convergence to safe regions of the MDP [92, 90], or optimize a policy to avoid constraint violations [3, 23, 91].

We wish to 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 [68]. However, unlike prior work [30, 80, 83, 84, 85, 101], 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.

2.2 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 [69]. DAgger [70] achieves sublinear regret by repeatedly soliciting human action labels in an online fashion. While there has been work on safe variants of DAgger [100, 33], 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 [4], 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 [1, 103, 25, 11]. Risk-sensitive IRL methods [44, 51, 75] 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 [64, 14, 12].

2.3 Sim2Real Transfer in Robot Learning

There is significant prior work on learning policies in simulation and facilitating transfer to physical experiments. The most common approach is domain randomization [49, 93], which varies dynamical and/or visual properties such as friction, lighting, camera angle, colors, and textures in simulation to enable zero-shot transfer to the physical world. Valassakis et al. [95] investigate dynamics domain randomization via injecting random forces into simulations. Domain randomization has achieved transfer in a variety of applications such as robot legged locomotion [46], fabric manipulation [35], and robotic grasping [49, 93]. Recent work has also explored adaptive methods and curricula for domain randomization, such as automatic domain randomization for dexterous manipulation of a Rubik's cube [57] and active domain randomization [54]. Other proposed approaches for crossing the reality gap include learning a canonical intermediate representation [39], domain adaptation via a small amount of real data [88], and "real2sim2real" tuning of simulation parameters based on real data [47, 65, 20]. Despite the significant body of work on simulated training procedures to enable physical deployment, there has been very little research on automatically determining precisely when to stop training in simulation and deploy learned policies in the physical world without any real evaluation. The closest prior work to ours may be Muratore et al. 2021 [55], who propose early stopping in simulation based on an upper confidence bound on the optimality gap during domain randomization. However, while this work is specific to the reinforcement learning paradigm, we apply our approach to the imitation learning setting, in which algorithms learn from

demonstrations.

2.4 Model Predictive Control

Model predictive control (MPC) and model-based reinforcement learning methods (MBRL) have seen success in robotics [53]. MPC iteratively exploits a model of a system to make predictions of future states of the system in order to plan actions to maximize reward.

Chua et al. 2018 [16] is a MBRL method which uses the cross-entropy method (CEM) to sample actions close to other previous actions which yielded high rewards and MPC for planning and optimizing trajectories.

While many safety-focused and constraint-aware versions of CEM and MPC have been studied [90, 45, 97, 96], none focus on planning safely when there is epistemic uncertainty over the reward function. Most of the prior work is robust to noise, with none being robust to uncertainty over the true reward to optimize for. Serving as one example, Wen et al. 2018 [97] impose polytopic constraints on states and actions. However, the constraints must be engineered for each scenario and explicit constraints may not always be precisely known. This gives a similar problem to human-defined reward functions, since the human must create constraints they believe explain what they desire the system to do.

Chapter 3

Preliminaries

3.1 Markov Decision Processes

We model the environment as a Markov Decision Process (MDP) [63]. 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 $\rho(\pi, r) = \mathbb{E}_{\tau \sim \pi_\theta}[r(\tau)]$.

For our study on stopping conditions for sim2real transfer, we consider learning a policy π for some task given access only to a computational simulation, which we model as a Partially Observable Markov Decision Process (POMDP) [40] $\mathcal{M}^{\phi}_{\text{sim}}$ parametrized by simulation parameters ϕ which captures the parameters of the POMDP (eg. visual appearance and dynamics of the simulation). The objective is then to achieve high performance at this task when this policy is deployed in the physical world, which we model as MDP $\mathcal{M}_{\text{real}}$.

More formally, we assume that both $\mathcal{M}_{\text{sim}}$ and $\mathcal{M}_{\text{real}}$ have shared state space $\mathcal{S}$, observation space $\mathcal{O}$, action space $\mathcal{A}$, reward function $R : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$, initial state distribution μ, and time horizon H, but may have different mappings from states $s \in \mathcal{S}$ to observations $o \in \mathcal{O}$. In this work, we consider grayscale image observations ($\mathcal{O} = \mathcal{R}^{H \times W \times C}$, $C = 1$), but consider settings (as is typical in practice) where observations corresponding to a specific state may have different visual appearance between simulation ($\mathcal{M}_{\text{sim}}$) and reality ($\mathcal{M}_{\text{real}}$). We additionally consider settings in which the state transition dynamics associated with $\mathcal{M}_{\text{sim}}$, denoted by $P_{\text{sim}} : \mathrm{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ may be different from that associated with $\mathcal{M}_{\text{real}}$, denoted by $P_{\text{real}} : \mathrm{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. This reflects the inability of computational simulations to

precisely model the dynamics of the physical world.

We consider a setup where robot policy $\pi : \mathcal{S} \to \mathcal{A}$ is first learned with some policy search algorithm using T_{sim} total transitions in simulated environment $\mathcal{M}_{\text{sim}}$ to optimize the following objective based on the attained rewards in simulation, where ϕ denotes the parameters of the simulation:

$$J^{\phi}_{\text{sim}}(\pi) = \mathbb{E}_{\pi, \mathcal{M}^{\phi}_{\text{sim}}} \sum_{t=1}^{H} R(o_t, a_t) \tag{3.1}$$

where the expectation is with respect to observation-action trajectories sampled from policy π in MDP $\mathcal{M}^{\phi}_{\text{sim}}$. The objective in this work is to identify the optimal stopping time T_{sim} such that when policy π is evaluated in the physical world, it maximizes the following objective based on the attained rewards in physical trials:

$$J_{\text{real}}(\pi) = \mathbb{E}_{\pi, \mathcal{M}_{\text{real}}} \sum_{t=1}^{H} R(o_t, a_t) \tag{3.2}$$

where analogously, the expectation is with respect to observation-action trajectories sampled from policy π in MDP $\mathcal{M}_{\text{real}}$.

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 [1, 103, 74] or as a deep neural network [31, 25]. 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 [98]. Alternatively, the distribution could be the posterior distribution $\mathbb{P}(R \,|\, D)$ learned via Bayesian inverse reinforcement learning [64] given demonstrations D, the posterior distribution $\mathbb{P}(R \,|\, \mathcal{P}, D)$ given preferences $\mathcal{P}$ over demonstrations [74, 12], or the posterior distribution $\mathbb{P}(R \,|\, r')$ learned via inverse reward design given a human-specified proxy reward r' [29, 66]. This distribution is typically only available via sampling techniques such as Markov chain Monte Carlo (MCMC) sampling [64, 29, 12].

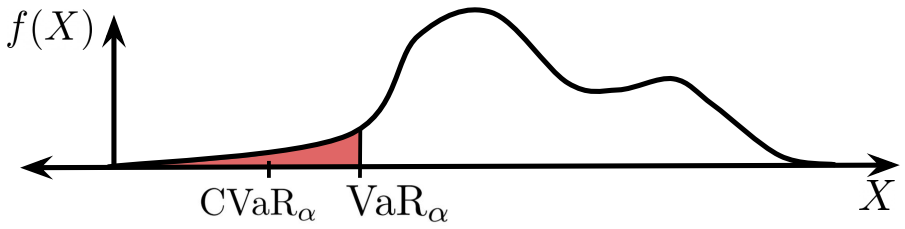


Figure 3.1: The pdf $f(X)$ of a random variable X. VaR_α measures the $(1-\alpha)$-quantile outcome. CVaR_α measures the expectation given that we only consider values less than the VaR_α.

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, $\rho(\pi, R), R \sim \mathbb{P}(R)$. We seek a robust policy that minimizes tail risk, given some risk measure, under the induced distribution ρ. Figure 3.1 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, $\rho(\pi, R)$, of a policy π under the reward function random variable R, and the objective is to minimize the tail risk (visualized in red).

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:

$$\text{VaR}_\alpha[X] = \sup\{x : \mathbb{P}(X \geq x) \geq \alpha\}, \tag{3.3}$$

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 [17], (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 [5].

Conditional Value at Risk

CVaR is a coherent risk measure [18], also known as average value at risk, expected tail risk, or expected shortfall. For continuous distributions

$$\text{CVaR}_\alpha[X] = \mathbb{E}_{f(X)}\left[X \mid X \leq \text{VaR}_\alpha[X]\right]. \tag{3.4}$$

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 [64, 74, 29, 13], we cannot directly optimize for CVaR using the definition in Equation (3.4) since this definition only works for atomless distributions. Instead, we make use of the following definition of CVaR, proposed by Rockafellar et al. [68], that works for any distribution:

$$\text{CVaR}_\alpha[X] = \max_\sigma \left(\sigma - \frac{1}{1-\alpha}\mathbb{E}[(\sigma - X)_+]\right), \tag{3.5}$$

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

$$\begin{aligned}
\text{CVaR}_\alpha[X] &= \mathbb{E}_{f(X)}[X \mid X \leq \sigma] && (3.6)\\
&= \sigma - \mathbb{E}_{f(X)}[\sigma - X \mid X \leq \sigma] && (3.7)\\
&= \sigma - \frac{\mathbb{E}_{f(X)}[\mathbf{1}_{X\leq\sigma} \cdot (\sigma - X)]}{P(X \leq \sigma)} && (3.8)\\
&= \sigma - \frac{1}{1-\alpha}\mathbb{E}_{f(X)}[(\sigma - X)_+] && (3.9)
\end{aligned}$$

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 $\text{VaR}_\alpha[X]$, and $(x)_+$. Taking the maximum over $\sigma \in \mathbb{R}$, gives us the definition in Equation (3.5).

Chapter 4

Policy Gradient Bayesian Robust Optimization for Imitation Learning (PG-BROIL)

Work done with Daniel Brown, Satvik Sharma, Jerry Zhu, and Ashwin Balakrishna.

4.1 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 $R \sim \mathbb{P}(R)$, [10] recently proposed Bayesian Robust Optimization for Imitation Learning (BROIL) which seeks to optimize the following:

$$\max_{\pi_\theta} \lambda \cdot \mathbb{E}_{\mathbb{P}(R)}[\psi(\pi_\theta, R)] + (1-\lambda) \cdot \text{CVaR}_\alpha \left[\psi(\pi_\theta, R)\right] \tag{4.1}$$

For MDPs with discrete states and actions and known dynamics, Brown et al. 2020 [10] 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. 2020 [12] considered two common metrics: (1) expected value, i.e., $\psi(\pi_\theta, R) = \rho(\pi, R) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$ and (2) baseline regret, i.e., $\psi(\pi_\theta, R) = \rho(\pi_\theta, R) - \rho(\pi_E, R)$ where π_E denotes an expert policy (usually estimated from demonstrations). For simplicity, we let $\psi(\pi_\theta, R) = \rho(\pi, R)$ (expected return) hereafter.

To find the policy that maximizes Equation (4.1) we need the gradient with respect to the policy parameters θ. For the first term in Equation (4.1), we have

$$\nabla_\theta \mathbb{E}_{\mathbb{P}(R)}[\rho(\pi_\theta, R)] \approx \sum_{i=1}^{N} \mathbb{P}(r_i) \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)]. \tag{4.2}$$

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

$$\nabla_\theta \max_\sigma \Big(\sigma - \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbb{P}(r_i) \big(\sigma - \mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)]\big)_+\Big) \tag{4.3}$$

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 $\rho(\pi, r_i)$, for each reward function sample from the posterior since these are the endpoints of each linear segment. If we let $\rho_i = \rho(\pi, r_i)$ then we can quickly iterate over all reward function hypotheses and solve for σ as

$$\sigma^* = \underset{\sigma\in\{\rho_1,\ldots,\rho_N\}}{\operatorname{argmax}} \Big(\sigma - \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbb{P}(r_i) \big[\sigma - \rho_i\big]_+\Big). \tag{4.4}$$

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

$$\rho_i \approx \frac{1}{|\mathcal{T}|} \sum_{\tau\in\mathcal{T}} \sum_{t=0}^{T} r_i(s_t, a_t). \tag{4.5}$$

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 (4.4), we perform a step of policy gradient optimization by following the sub-gradient of CVaR with respect to the policy parameters θ:

$$\nabla_\theta \, \text{CVaR}_\alpha = \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbb{P}(r_i) \mathbf{1}_{\sigma^* \geq \rho(\pi_\theta, r_i)} \nabla_\theta \rho(\pi_\theta, r_i) \tag{4.6}$$

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 (4.6), the only thing remaining to compute is the standard policy gradient. Note that in standard RL, we write the policy gradient as [81]:

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t(\tau) \right] \tag{4.7}$$

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 [76] with respect to some single reward function:

$$\Phi_t(\tau) = A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t). \tag{4.8}$$

We define $\Phi_t^{r_i}$ as some measure of the quality of the policy in terms of a particular 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) [76]:

$$\Phi_t^{r_i} = A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t). \tag{4.9}$$

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 $\tau \sim \pi_\theta$:

$$\nabla_\theta \text{BROIL} = \sum_i \mathbb{P}(r_i) \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[r_i(\tau)] \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.10}$$

$$= \sum_i \mathbb{P}(r_i) \left(\mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \right] \right) \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.11}$$

$$\approx \sum_i \mathbb{P}(r_i) \left(\frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \right] \right) \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.12}$$

$$= \frac{1}{|\mathcal{T}|} \sum_i \mathbb{P}(r_i) \left(\sum_{\tau \in \mathcal{T}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \right] \right) \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.13}$$

$$= \frac{1}{|\mathcal{T}|} \sum_i \sum_{\tau \in \mathcal{T}} \mathbb{P}(r_i) \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \right] \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.14}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_i \mathbb{P}(r_i) \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \right] \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.15}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_i \sum_{t=0}^{T} \mathbb{P}(r_i) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.16}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} \sum_i \mathbb{P}(r_i) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.17}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(\sum_i \mathbb{P}(r_i) \Phi_t^{r_i}(\tau) \big(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \big) \right) \tag{4.18}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) w_t \tag{4.19}$$

where

$$w_t(\tau) = \sum_{i=1}^{N} \mathbb{P}(r_i) \Phi_t^{r_i}(\tau) \left(\lambda + \frac{1-\lambda}{1-\alpha} \mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)} \right) \tag{4.20}$$

Algorithm 1 Policy Gradient BROIL

Input: initial policy parameters θ_0, samples from reward function posterior $r_1, \ldots, r_N$ and associated probabilities, $\mathbb{P}(r_1), \ldots, \mathbb{P}(r_N)$.
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{T}_k = \{\tau_i\}$ by running policy π_{θ_k} in the environment.
  Estimate expected return of π_{θ_k} under each reward function hypothesis r_j using Eq. (4.5).
  Solve for σ^* using Eq. (4.4)
  Estimate policy gradient using Eq. (4.19) and Eq. (4.20).
  Update θ using gradient ascent.
end for

is the weight associated with each state-action pair (s_t, a_t) in the set of trajectory rollouts $\mathcal{T}$.

If $\lambda = 1$, then

$$w_t(\tau) = \sum_{i=1}^{N} \mathbb{P}(R_i)\Phi_t^{R_i}(\tau) = \Phi_t^{\bar{R}}(\tau) \tag{4.21}$$

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

$$w_t(\tau) = \frac{1}{1-\alpha} \sum_{i=1}^{N} \mathbf{1}_{\sigma^* \geq \rho(\pi, R_i)} \mathbb{P}(R_i)\Phi_t^{R_i}(\tau) \tag{4.22}$$

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 $\rho(\pi, R_i) \leq \sigma^*$, 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.

The PG-BROIL algorithm is shown in Algorithm 1.

4.3 BROIL Policy Gradient with Entropic Risk Measure

Another common risk metric, Entropic Risk Measure (ERM) [24], is also 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 [56] and soft-robust optimization with respect to model uncertainty [72].

Entropic Risk Measure

The entropic risk measure [24] is another form of tail risk that has the benefit of being everywhere differentiable, eliminating the need for subgradients as we saw in the derivation for CVaR. The entropic risk measure (ERM) of a random variable X is defined as:

$$ERM = -\frac{1}{\alpha}\log \mathbb{E}[e^{-\alpha X}] \tag{4.23}$$

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

We start with the objective:

$$\underset{\pi_\theta}{\text{maximize}} \quad \lambda \cdot \mathbb{E}[\psi(\pi_\theta, R)] + (1-\lambda)\cdot \text{ERM}_\alpha\Big[\psi(\pi_\theta, R)\Big] \tag{4.24}$$

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

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

$$\nabla_\theta \cdot \mathbb{E}_{\mathbb{P}(R)}[\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]] = \sum_i \mathbb{P}(r_i)\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r_i(\tau)]. \tag{4.25}$$

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

$$\nabla_\theta \text{ERM}_\alpha[\rho(\pi, R)] = -\nabla_\theta \frac{1}{\alpha} \log \left(\sum_i \mathbb{P}(r_i) e^{-\alpha \rho(\pi_\theta, r_i)} \right) \tag{4.26}$$

$$= -\frac{1}{\alpha} \frac{1}{\sum_j \mathbb{P}(R_j) e^{-\alpha \rho(\pi_\theta, R_j)}} \sum_i \mathbb{P}(r_i) \nabla_\theta e^{-\alpha \rho(\pi_\theta, r_i)} \tag{4.27}$$

$$= -\frac{1}{\alpha} \frac{1}{\sum_j \mathbb{P}(R_j) e^{-\alpha \rho(\pi_\theta, R_j)}} \sum_i \mathbb{P}(r_i) e^{-\alpha \rho(\pi_\theta, r_i)} \nabla_\theta (-\alpha \rho(\pi_\theta, r_i)) \tag{4.28}$$

$$= \sum_i \frac{\mathbb{P}(r_i) e^{-\alpha \rho(\pi_\theta, r_i)}}{\sum_j \mathbb{P}(R_j) e^{-\alpha \rho(\pi_\theta, R_j)}} \nabla_\theta \rho(\pi_\theta, r_i) \tag{4.29}$$

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 $\tau \sim \pi_\theta$:

$$\rho(\pi_\theta, R_j) = \mathbb{E}_{\tau \sim \pi_\theta}[r_i j(\tau)] \approx \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} R_j(\tau) = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} R_j(s_t, a_t). \tag{4.30}$$

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:

$$\nabla_\theta \text{BROIL} = \lambda \sum_i \mathbb{P}(r_i) \nabla_\theta \rho(\pi_\theta, r_i) + (1-\lambda) \sum_i \frac{\mathbb{P}(r_i) e^{-\alpha \rho(\pi_\theta, r_i)}}{\sum_j \mathbb{P}(R_j) e^{-\alpha \rho(\pi_\theta, R_j)}} \nabla_\theta \rho(\pi_\theta, r_i) \tag{4.31}$$

$$= \sum_i \mathbb{P}(r_i) \nabla_\theta \rho(\pi_\theta, r_i) \left(\lambda + (1-\lambda) \frac{e^{-\alpha \rho(\pi_\theta, r_i)}}{\mathbb{E}_{\mathbb{P}(R)}[e^{-\alpha \rho(\pi_\theta, R)}]} \right) \tag{4.32}$$

As before we can write the policy gradient as

$$\nabla_\theta \rho(\pi_\theta, r_i) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[r_i(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \right]. \tag{4.33}$$

Defining $\Phi_t^{r_i}$ in terms of a particular reward function hypothesis r_i and approxi-

mating expectations with a set $\mathcal{T}$ of on-policy trajectories $\tau \sim \pi_\theta$ gives:

$$\nabla_\theta \text{BROIL} \approx \sum_i \mathbb{P}(r_i) \left(\frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \right] \right) \left(\lambda + (1-\lambda) \frac{e^{-\alpha \rho(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha \rho(\pi_\theta, R)}]} \right) \tag{4.34}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} \sum_i \mathbb{P}(r_i) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Phi_t^{r_i} \left(\lambda + (1-\lambda) \frac{e^{-\alpha \rho(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha \rho(\pi_\theta, R)}]} \right) \tag{4.35}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(\sum_i \mathbb{P}(r_i) \Phi_t^{r_i}(\tau) \left(\lambda + (1-\lambda) \frac{e^{-\alpha \rho(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha \rho(\pi_\theta, R)}]} \right) \right) \tag{4.36}$$

$$= \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) w_t \tag{4.37}$$

where

$$w_t = \sum_i \mathbb{P}(r_i) \Phi_t^{r_i}(\tau) \left(\lambda + (1-\lambda) \frac{e^{-\alpha \rho(\pi_\theta, r_i)}}{\mathbb{E}_R[e^{-\alpha \rho(\pi_\theta, R)}]} \right) \tag{4.38}$$

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. 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 $\rho(\pi_\theta, r_i)$ 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.

4.4 Trust Region PG-BROIL (PPO)

We also derive a version of the Proximal Policy Optimization (PPO) [77] 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:

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{(s,a)\sim\pi_{\theta_k}}[L(a, s, \theta_k, \theta)] \tag{4.39}$$

where

$$L(a, s, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right) \tag{4.40}$$

and

$$g(\epsilon, A^{\pi_{\theta_k}}(s, a)) = \begin{cases} (1+\epsilon)A^{\pi_{\theta_k}}(s, a) & A^{\pi_{\theta_k}}(s, a) \geq 0 \\ (1-\epsilon)A^{\pi_{\theta_k}}(s, a) & A^{\pi_{\theta_k}}(s, a) < 0 \end{cases} \tag{4.41}$$

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

$$w_t = \sum_i \mathbb{P}(r_i)\Phi_t^{r_i}(\tau)\big(\lambda + \frac{1-\lambda}{1-\alpha}\mathbf{1}_{\sigma^* \geq \rho(\pi, r_i)}\big) \tag{4.42}$$

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

The PPO-clip BROIL algorithm is written in Algorithm 2.

Algorithm 2 PPO-clip BROIL

1: **Input:** initial policy parameters θ_0, samples from reward function posterior $R_1, \ldots, R_N$ and associated probabilities, $\mathbb{P}(R_1), \ldots, \mathbb{P}(R_N)$, and any form for policy gradient weights Φ_t
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{T}_k = \{\tau_i\}$ by running policy π_θ in the environment.
4: Estimate expected return of π_θ under each reward function hypothesis r_j using Eq. (4.5).
5: Solve for σ^* using Eq. (4.4)
6: Update θ with stochastic gradient ascent by maximizing the PPO-clip objective:

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}} \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\left[\frac{1}{T}\sum_{t=0}^{T}\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta k}(a|s)}w_t, g(\epsilon, w_t)\right)\right]$$

using Eq. (4.42) for w_t.
7: **end for**

Chapter 5

PG-BROIL Experiments

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 [9], a pointmass navigation task inspired by [92] and a robotic reaching task from the from the DM Control Suite [89]. All domains are characterized by an agent 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 [2]. For CartPole we implement PG-BROIL on top of REINFORCE [59], using Algorithm 1 and for remaining domains we implement PG-BROIL on top of PPO [77], using Algorithm 2.

Experimental Domains

CartPole: We consider a risk-sensitive version of the classic CartPole benchmark [9]. 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 5.1a. 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 Thananjayen et al. 2020 [92]. 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 5.1b. 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:

$$R(s) = -\left(\|s_{x,y} - g\|_2^2 + b \cdot \mathbf{1}_{gray}\right), b \sim \mathbb{P}(b), \tag{5.1}$$

where $\mathbf{1}_{gray}$ 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 [89] (Figure 5.1c), 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 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

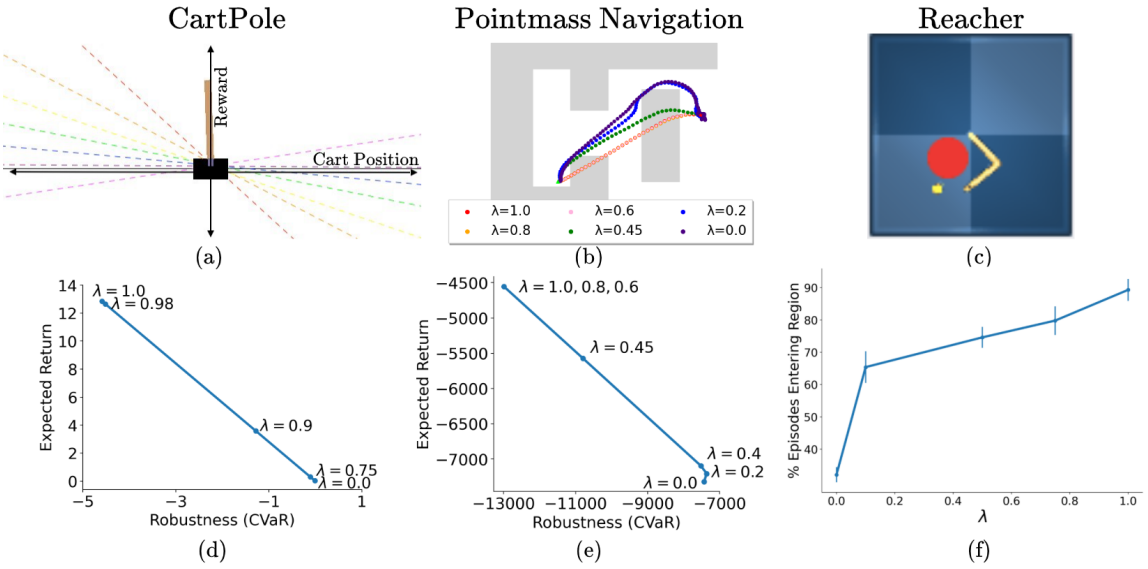


Figure 5.1: **Prior over Reward Functions: Domains and Results.** We study (a) CartPole in which the reward is an unknown linear function of the cart's position, (b) Pointmass Navigation with gray regions of uncertain costs, and (c) Reacher with a red region of uncertain cost. For the CartPole and Pointmass 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.

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).

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 5.1d and 5.1e respectively. Figure 5.1e 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 5.1b. 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 5.1f).

5.2 Learning From Human and Artificial Demonstrations

We now consider the imitation learning setting, where an agent 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 [25, 22, 11] will output only one of them—not necessarily the true reward. There has been some work on Bayesian algorithms such as Bayesian IRL [64] which estimates a *posterior distribution* instead of a single reward and Bayesian REX [12] 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 [64, 14, 12]. 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 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.

TrashBot from Human Demonstrations

We first consider a continuous control TrashBot domain (Figure 5.2), 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 contributors to PG-BROIL) as shown in Figure 5.2. 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 [42]. We hypothesize that PG-BROIL will hedge against uncertainty and learn to pick up trash while avoiding the gray region.

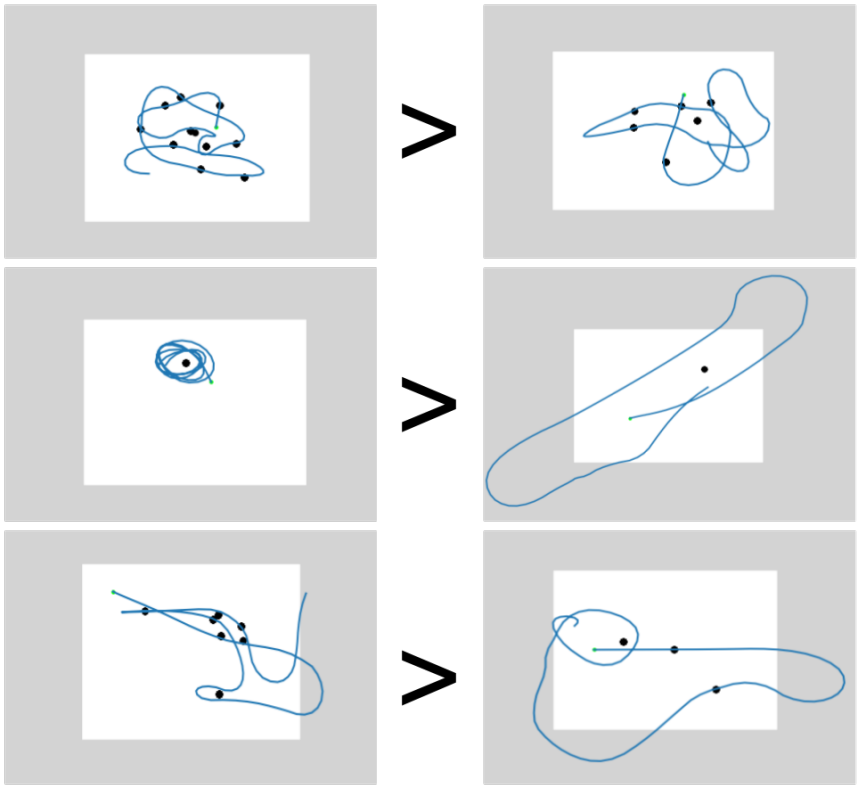


Figure 5.2: **TrashBot environment:** Each time the robot picks up a piece of trash (by moving close to a black dot), a new one appears at a randomly in the white region. We give pairwise preferences over human demos that aim to teach the robot that picking up trash is good (left), going into the gray region is undesirable (center), and less time in the gray region and picking up more trash is preferred (right).

We compare against behavioral cloning (BC), GAIL [31], and Risk-Averse Imitation Learning (RAIL) [75], 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) [15] in the offline demonstration setting [11] which optimizes an MLE estimate of the reward weights, and Bayesian REX [12], which optimizes the mean reward function under the posterior distribution given the preferences. PG-BROIL also uses Bayesian REX [12] to infer a

Table 5.1: **TrashBot:** We evaluate PG-BROIL against 5 other imitation learning algorithms when learning from ambiguous preferences over demonstrations (Figure 5.2). 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	**8.4 ± 0.5**	**0.1 ± 0.1**

reward function posterior distribution given the preferences over demonstrations, but optimizes the BROIL objective.

Table 5.1 compares the performance of each baseline imitation learning algorithm when given the 3 pairs of demonstrations shown in Figure 5.2. We find that PG-BROIL outperforms BC and GAIL [31] 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 also outperforms PBRL and Bayesian REX, which overfit to staying in the white region, not realizing the importance of picking trash.

Figure 5.3 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 and Bayesian REX, which fall into a local maxima by simply mining rewards by staying in the white region. 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

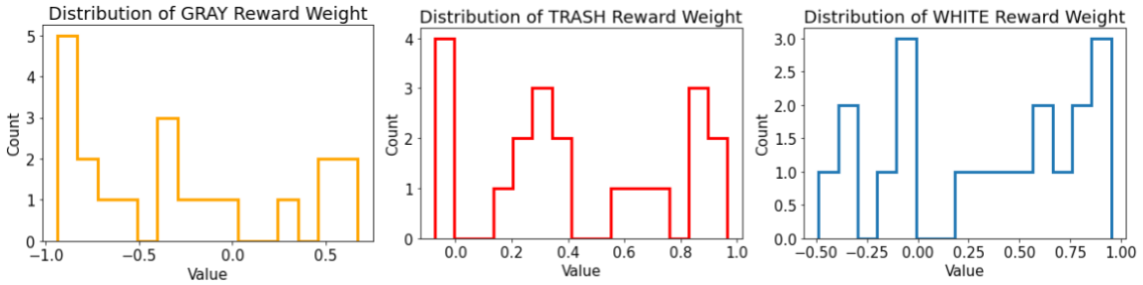
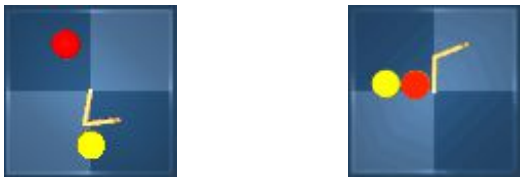


Figure 5.3: Reward distribution generated by Bayesian REX of each feature weight in posterior for seed 0.

vice-versa. This helps create the causal confusion we see in this experiment since it is unclear whether the agent should be rewarded more for the WHITE feature or the TRASH feature.

Reacher from Demonstrations 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 environment before and after the domain shift is shown in Figure 5.4a The inferred reward function is a linear combination of 2 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 5.2. By contrast, PG-BROIL hedges against its uncertainty over the quality of the uncertain region and avoids it.

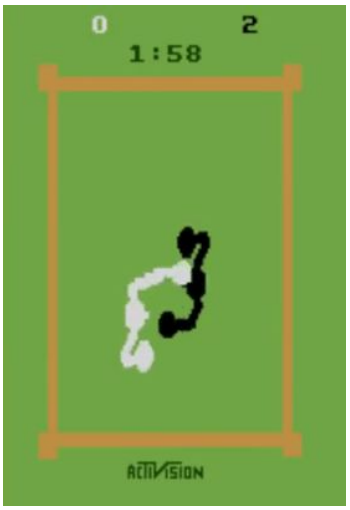


(a) (b)

Figure 5.4: 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.

Table 5.2: **Reacher from Demos:** We evaluate PG-BROIL and baseline imitation learning algorithms when learning from preferences over demonstrations. Results are averages ($\pm$ 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 $\pm$ 27.4	39.9 $\pm$ 62.3
GAIL	2.3 $\pm$ 1.7	5.1 $\pm$ 13.0
RAIL	2.1 $\pm$ 1.2	4.6 $\pm$ 27.0
PBRL	28.4 $\pm$ 37.7	16.8 $\pm$ 30.4
Bayesian REX	13.5 $\pm$ 35.0	94.5 $\pm$ 70.1
PG-BROIL	**1.7 $\pm$ 7.2**	**102.0 $\pm$ 60.5**



(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 5.5: **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).

Atari Boxing from Demonstrations

For this experiment, we give the agent 3 preferences over suboptimal demos of the Atari Boxing game [6]. 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 5.5b.

Chapter 6

BROIL with Cross Entropy Method and Model Predictive Control (MPC-BROIL)

Work done with Daniel Brown, Satvik Sharma, and Rishi Parikh.

6.1 Algorithm

This method maximizes the soft robust BROIL objective within a CEM and MPC approach for model-based planning. In model-based planning, CEM is

The main change from the normal CEM and MPC procedure lies in the CEM portion. At each iteration of CEM, after randomly picking action samples from a Gaussian fit to the elites, the dynamics model is used to determine the rewards under each reward function. Then, T_{best} is created according to Equation 6.1 and this set updates the elites for the next iteration of CEM. After CEM has completed, the next action, a_0 from the best action trajectory τ_{max} is selected and executed in the environment.

$$T_{best} := \{\tau_i \,|\, \forall i \in [1...N_{elite}]\} \tag{6.1}$$

Where τ_i is determined by:

$$\tau_i = \underset{\tau \in T_c}{argmax_i} \lambda \cdot \mathbb{E}[\psi(\tau, R)] + (1-\lambda) \cdot \text{CVaR}_\alpha[\psi(\tau, R)]\} \tag{6.2}$$

and $\underset{\tau \in T_c}{argmax_i}$ is the τ corresponding to the i^{th} largest value.

The full MPC-BROIL algorithm is written in detail in Algorithm 3.

Algorithm 3 MPC-BROIL

Input: reward function posterior $r_1, \ldots, r_N$ and associated probabilities, $\mathbb{P}(r_1), \ldots, \mathbb{P}(r_N)$, horizon h, the number of elites N_{elite}, the number of CEM iterations N_{CEM}, dynamics model parameters θ,
for $k = 0, 1, 2, \ldots$ **do**
  for $c = 0, 1, 2, \ldots N_{CEM}$ **do**
    if k == 0 then **then**
      Collect trajectories $\mathcal{T}_0 = \{\tau_i\}$ by randomly selecting actions from the action space.
    else
      Collect trajectories $\mathcal{T}_c = \{\tau_i\}$ by randomly selecting actions from a multivariate normal distribution with mean μ_h and a diagonal covariance matrix $\Sigma_{h \times h}$.
      for each candidate trajectory i **do**
        With the dynamics model θ rollout τ_i under each reward function r_j to calculate the reward estimate R_i
        Calculate the $Cvar_i$ and σ_i values using R_i and $\mathbb{P}(r_1), \ldots, \mathbb{P}(r_N)$
        Calculate B_i using Eq. 17
      end for
    end if
    Select the best N_{elite} trajectories $\mathcal{T}_{best} = \{\tau_i\}$ using the corresponding values of B_i
    Use $\mathcal{T}_{best}$ to update μ_h and $\Sigma_{h \times h}$.
  end for
  Return first action from best τ_{max} in $\mathcal{T}_{N_{CEM}}$
end for

Chapter 7

MPC-BROIL Experiments

7.1 Prior Distribution over Rewards and Known Dynamics

We test whether we can still get a family of different behaviors by optimizing the BROIL objective via MPC. To test this out we first simplify the problem by assuming a perfect dynamics model to reduce the noise from a learned dynamics model and also assume a prior distribution over costs, avoiding the need to learn a distribution over reward functions.

We test using a simple Pointmass environment. This is the exact same environment as used in the PG-BROIL experiment described in Section 5.1. We also use the exact same reward function posterior.

The results are shown in Figure 7.1. We see that for different values of λ we get different qualitative behavior. For $\lambda = 1$ the agent goes almost directly to the goal, and for $\lambda = 0$ the agent avoids the gray areas. For $0 < \lambda < 1$ we see behavior that partially avoids the gray area, hedging against the possibly high costs of the gray region with the known cost of distance to the goal.

As we saw with PG-BROIL, we see that MPC-BROIL also leads to a range of different behaviors based off of different values for the hyperparameter λ.

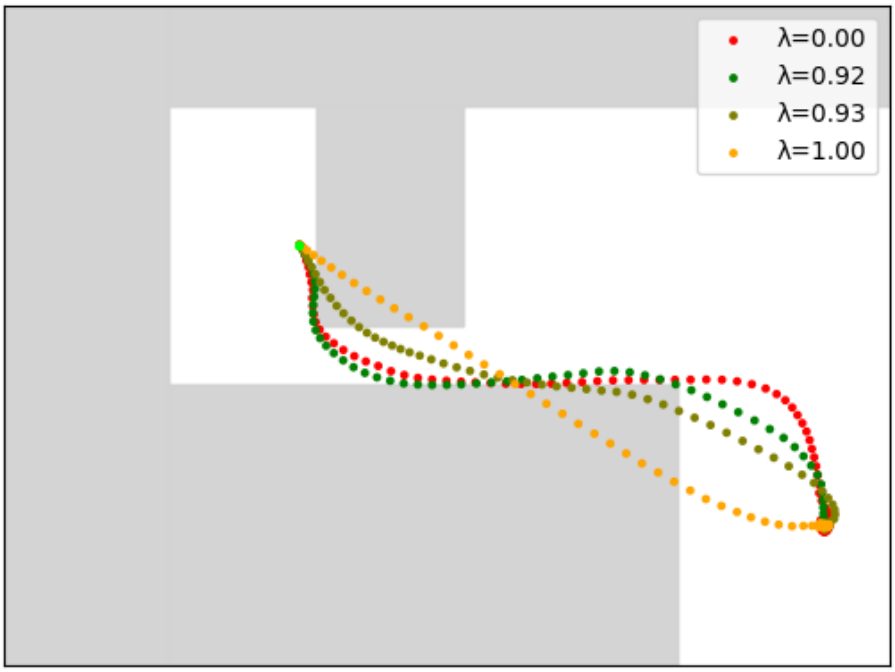


Figure 7.1: **MPC-BROIL in Pointmass Navigation:** We evaluate MPC-BROIL on a Pointmass environment. An oracle dynamics model was used with a planning horizon of 20.

Chapter 8

Switching Criteria for Imitation Learning

Work done with Ellen Novoseller, Satvik Sharma, Vainavi Viswanath, Rishi Parikh, Ryan Hoque, Ashwin Balakrishna, and Daniel Brown.

8.1 Algorithm Description

We consider switching from simulation to real in the case of imitation learning, in which an algorithm learns a policy from sequentially-collected demonstrations. We propose four criteria to determine when to switch from simulated training to physical deployment. Specifically, we will describe two evaluation metrics for switching (reward in simulation and epistemic uncertainty) and two stopping conditions for choosing when to switch (absolute threshold and gradient-based). Pairing the two evaluation metrics and two stopping conditions results in four possible switching criteria, each a combination of a particular switching metric and stopping condition.

To study when to switch from learning in simulation to deploying policies in physical experiments, we learn a switching criterion $\psi : \pi \to \{0, 1\}$. If $\psi(\pi) = 1$, we terminate simulated training and deploy π in physical experiments, while otherwise, we continue training π. The switching function ψ identifies a switching time T_{sim} as a function of the evolving robot policy π.

In the imitation learning setting, the objective is to learn a robot policy π that emulates task demonstrations collected in simulation from some demonstration policy π_D. In principle, π_D could be a human controlling the robot in simulation via teleoperation or an algorithmic controller defined using privileged state information present only in simulation; we consider the latter. We train π in simulation with

behavior cloning, in which the objective is to minimize the following loss function to encourage the robot's policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ to match that of the demonstrator (π_D) on a set of trajectories $\mathcal{D}$ collected from the demonstrator:

$$J_{BC}(\pi) = \mathbb{E}_{(s_t, \pi_D(s_t)) \sim \mathcal{D}} \left[\mathcal{L}(\pi(s_t), \pi_D(s_t)) \right] \tag{8.1}$$

where $\mathcal{L}(\pi(s), \pi_D(s))$ is an action discrepancy measure between $\pi(s)$ and $\pi_D(s)$ (e.g., the squared loss or 0-1 loss; our experiments use the squared loss).

The key objective is to determine how many training iterations in simulation are required to train π before it should be deployed on a physical system. To this end, we train π to minimize the loss in Equation (8.1) in simulation on trajectories in $\mathcal{D}$ and compute a stopping criterion $\psi : \pi \rightarrow \{0, 1\}$. If $\psi(\pi) = 1$, we terminate simulated training and deploy π in physical experiments. Otherwise, we query π_D for an additional task demonstration in simulation, aggregate this demonstration into $\mathcal{D}$, and continue.

Algorithm 4 fully describes the process. The procedure alternates between collecting a batch of demonstrations from the demonstration policy π_D and appending the demonstrations to the dataset $\mathcal{D}$, performing a model update epoch to minimize the loss function $J_{BC}(\pi)$, and checking whether the switching criterion $\psi(\pi)$ is satisfied. If it is, then the process is halted and the final learned policy π is deployed on the physical robot system. In practice, our experiments collect a batch of $K = 10$ demonstrations per iteration, where we use $K = 10$ to obtain sufficient granularity of information for determining the switching point. Each model update epoch performs 400 gradient steps on the loss function $J_{BC}(\pi)$ with a minibatch size of 64. We optimize with Adam [41], with a learning rate of $2.5 * 10^{-4}$ and L2 regularization of 10^{-5}.

8.2 Switching Metrics

We propose two switching metrics, which are computed after every model update step in Algorithm 4.

Simulation Reward: Here we evaluate π in simulation and compute the average total reward over L rollouts to approximate $J_{\text{sim}}(\pi)$ from Equation (3.1). We also use this dataset to tune our switching conditions via cross-validation in simulation, as detailed below, and denote it as $\mathcal{D}_{cross}$. In practice, our experiments use $L = 5$, which we found to provide a set of rewards with sufficiently small standard error. Intuitively, performing well in simulation suggests that the policy may transfer well to real.

Algorithm 4 Learned Switching Criteria for Sim2Real

Input: Maximum number of episodes N; number of demonstrations K collected between each model update and calculation of the stopping criterion $\psi(\pi)$; demonstration policy π_D; initialized buffer for demonstration data $\mathcal{D} \leftarrow \emptyset$; randomly initialized robot policy π
for $i \in \{1, \dots N\}$ **do**
  Collect $\tau_j^{\text{demo}} = ((s_t, a_t)_{t=1}^T)$, $j \in \{1, \dots, K\}$ from π_D
  $\mathcal{D} = \mathcal{D} \cup \tau_1^{\text{demo}} \cup \dots \cup \tau_K^{\text{demo}}$
  $\pi \leftarrow \arg\min_\pi \mathbb{E}_{(s_t, \pi_D(s_t)) \sim \mathcal{D}} \left[\mathcal{L}(\pi_R(s_t), \pi_D(s_t))\right]$
  if $\psi(\pi)$ **then**
    Terminate learning and deploy on physical system
  end if
end for

Epistemic Uncertainty: Here we approximate the epistemic uncertainty of π to characterize the policy's confidence in the actions that it predicts. We estimate epistemic uncertainty by training an ensemble of E policies on bootstrapped minibatches of the training data in $\mathcal{D}$. We then estimate the epistemic uncertainty over a holdout set $\mathcal{D}_{\text{test}}$, which is not present during model training. Intuitively, the more consistent (low-variance) the policy's predictions on $\mathcal{D}_{\text{test}}$ are across ensemble members, the higher the probability that the policy has converged. Let $\{\pi^i\}_{i=1}^E$ denote the policy ensemble and $\{a_s^i\}_{i=1}^E \in \mathbb{R}^M$ denote the corresponding actions when each ensemble member is queried at observation s. Further, let a_{ij} denote the j^{th} component of the action predicted by ensemble member i. The epistemic uncertainty is estimated via:

$$\mathbb{E}_{s \in \mathcal{D}_{\text{test}}} \left[\frac{1}{M} \sum_j \text{Var}_i(a_{ij}) \right], \tag{8.2}$$

where $\text{Var}_i(a_{ij})$ denotes the variance over ensemble members $i \in \{1, 2, \dots E\}$ in action component j, and the expectation is taken over the observation-action pairs in $\mathcal{D}_{\text{test}}$.

8.3 Stopping Conditions

Here, we discuss two stopping conditions based on the metrics in the previous section for determining when to switch from simulation to real.

Value-Based: The first stopping condition is an absolute threshold, A. The cross-validation set, $\mathcal{D}_{cross}$, is used to tune A: for simulation reward, A is set to approximately the maximum reward attained in $\mathcal{D}_{cross}$ (on average over the L rollouts

in $\mathcal{D}_{cross}$), while for epistemic uncertainty, it is highest epistemic uncertainty that corresponds with the earliest reward peak in $\mathcal{D}_{cross}$. Specific values of A are provided in the supplemental website. At each evaluation episode i, a spline $f(x)$ is fit to the previous $i-1$ data points. We use a spline—rather than raw data—to mitigate the noise in the switching criteria. Then, the spline is evaluated at the current episode and checked against the predefined absolute threshold A to determine whether the algorithm should stop training at the current episode and deploy the policy in real or continue to learn in simulation.

Gradient-Based: The second stopping condition is gradient-based, and determines when the change over time in an evaluation metric falls below a threshold. Similar to the absolute threshold, we identify the threshold by fitting a spline to the previous $i-1$ data points. Then, finite differences are taken for two points evaluated on the spline to approximate the current gradient. If the gradient is within the range $[-\epsilon, \epsilon]$ (tuned for each switching metric as described below), the gradient is considered to be sufficiently close to zero and two counters are potentially incremented: w_{consec} keeps track of the number of consecutive episodes with a sufficiently-small gradient, while w_{total} keeps track of the total number of episodes with a below-threshold gradient. The stopping condition is triggered if either $w_{consec} > U$ or $w_{total} > V$, where U is the maximum limit of continuous evaluation episodes before stopping and V is the maximum limit of noncontinuous evaluation episodes. The parameter V allows training to stop even if there are some aberrations which would reset w_{consec} to zero. Furthermore, the V condition alone is insufficient, since if the gradient frequently oscillates near zero due to noise in the switching metric, then the check for V would trigger even though we may not have converged to acceptable performance, indicating that the policy can still learn and should not yet be deployed in real.

Fig 8.1 illustrates this process. In practice, the parameters (ϵ, U, V) of the stopping conditions are tuned via cross-validation in simulation, using the dataset $\mathcal{D}_{cross}$. In particular, ϵ is empirically set to the largest value such that the range $[-\epsilon, \epsilon]$ does not classify many earlier iterations (which have comparatively larger gradients) as having a sufficiently-small gradient. U and V are tuned in conjunction so that the stopping condition is triggered as early as possible but not too early (i.e., for an overly-low simulation reward).

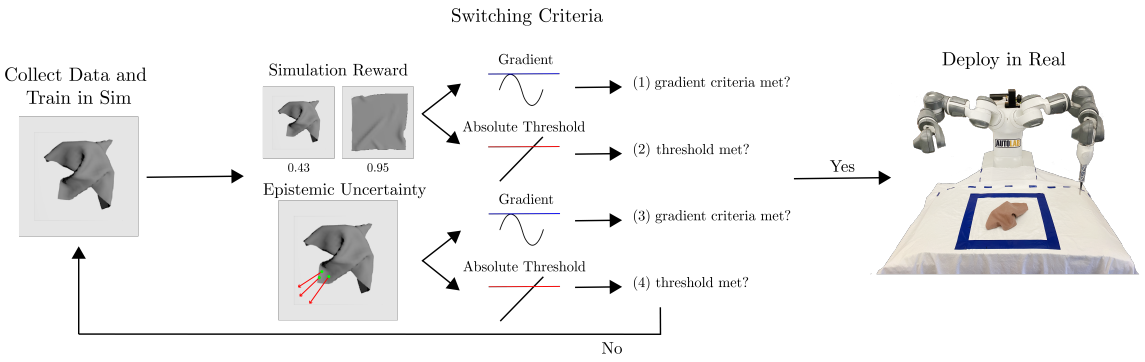


Figure 8.1: **System Overview.** At each step, our algorithm pipeline collects a new batch of simulation data, performs a model update epoch, and then checks whether a switching condition is satisfied. If the switching criterion is met, then the model is ready to be deployed in real. Otherwise, we continue collecting simulation data for further updating the model. We test four switching criteria, which utilize metrics based on a) reward when evaluated in simulation and b) epistemic uncertainty as estimated via an ensemble of policy networks, paired with each of two stopping conditions based on 1) absolute thresholding or values, and 2) gradients.

Chapter 9

Sim2Real Switching Criteria Experiments

9.1 Experimental Setup

Problem Statement

We consider the specific manipulation task of sequential fabric smoothing, a challenging open problem in robotics that has received significant recent interest [79, 34, 26]. As described in prior work [79], the objective in sequential fabric smoothing is to find a sequence of robot actions to maximally smooth a fabric from an initially crumpled configuration. We consider a square crop of fabric with initial configuration (state) s_0 and configuration s_t at time t. In both simulation and real, the robot can access only top-down grayscale image observations of the workspace, $\mathbf{o}_t \in \mathcal{O} = \mathbb{R}^{H \times W}$, where H and W are the image height and width respectively. Following prior work [33], we assume that each side of the fabric is monochromatic and that the two sides are colored differently, where the colors are distinguishable in grayscale (see supplement). At each timestep t, the robot executes a 4D pick-and-place action a_t parameterized by $(x_t, y_t, \Delta x_t, \Delta y_t)$, where (x_t, y_t) is the pick point in pixel space and $(x_t + \Delta x_t, y_t + \Delta y_t)$ is the place point. Through a known pixel-to-world transform, the robot picks and lifts the top layer of fabric at (x_t, y_t), translates by $(\Delta x_t, \Delta y_t)$, and releases. The robot seeks to learn a policy $\pi : \mathcal{O} \rightarrow \mathcal{A}$ that maximizes $R(s_T)$, where T denotes the final step of a fabric smoothing episode and $R(\cdot)$ gives the *coverage* or 2D area covered by the fabric. Following prior work [79], we assume that each smoothing episode terminates when the fabric reaches a coverage threshold of at least 92% or a limit of 10 actions (whichever happens first).

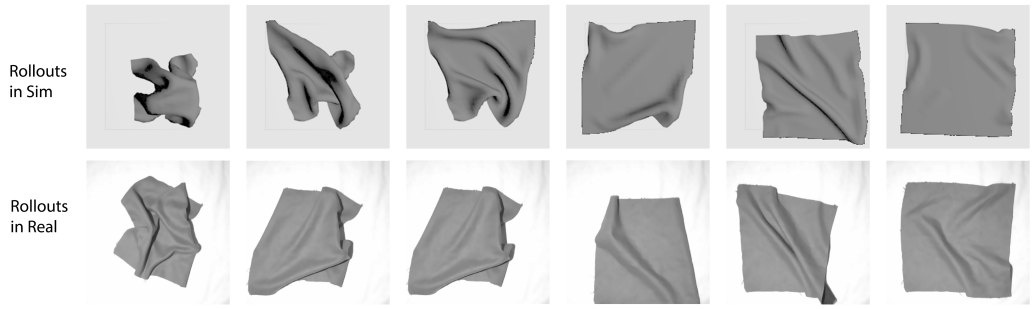


Figure 9.1: **Example Rollouts in Simulation and Physical Experiments.** The top row (left to right) depicts a sample trajectory in simulation, while the bottom row similarly depicts a sample physical robot trajectory.

Simulation Environment Setup

To collect demonstrations, we use the oracle corner-pulling policy in the Gym-Cloth simulator [79], which is ideal for fabric manipulation tasks. Gym-Cloth is an OpenAI-Gym style fabric manipulation environment, which simulates the cloth using structural, shear, and flexion springs. We use this simulator to collect data for training and testing our policies in simulation before switching to deployment in real. The cloth is represented as a 25x25 grid of point masses and is double-sided, with one side light blue and the other side dark blue. The images generated for learning are 224x224 px, of which a completely smooth cloth would occupy 164x164 px. This ratio is reflected in our physical set up. Lastly, when using the Gym-Cloth-simulated data to train fabric smoothing policies, we perform color shifts and augmentations. Specifically, the background color and the color of the two sides of the cloth are darkened, converted to gray scale, and blurred by a constant amount to match the images in real. Rollouts in sim and real in Fig 9.1 show the sim images converted to closely resemble real images. We collect a dataset $\mathcal{D}$ of 2,000 episodes of the oracle corner-pulling policies in Gym-Cloth for training behavior cloning policies.

Physical Environment Setup

The workspace contains a bimanual ABB YuMi robot with a single tweezer gripper on its left arm (the right arm is unused). Tweezer grippers are ideal for grasping the cloth as they are able to apply fine point pressure, which many standard grippers cannot do [79]. The manipulation surface is white and foam-padded to prevent end effector damage during collisions with the workspace. The experimental workspace is

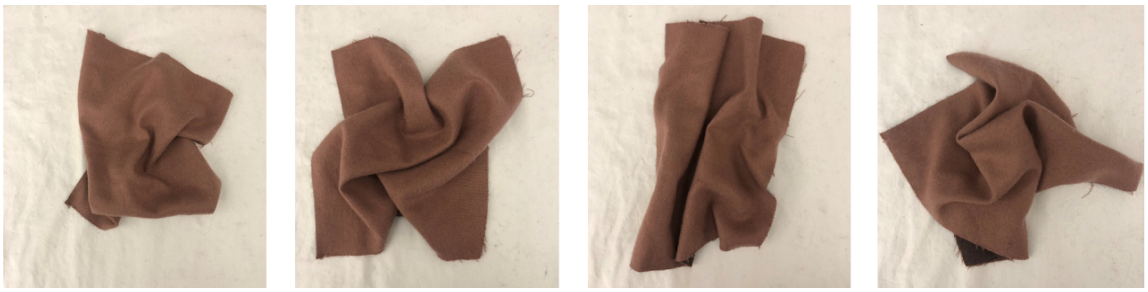


Figure 9.2: Four reset positions used in physical experiments. Each position is reset manually by the human at the start of every trajectory in order to make comparisons between different switching criteria as fair as possible.

marked with blue tape, so that the size of the manipulation workspace with respect to the size of the cloth resembles the set up in the Gym-Cloth simulator. We use a double sided 25 cm x 25 cm cloth where one side is light brown while the other side is dark brown. The workspace has an overhead PhotoNeo Phoxi Camera that captures grayscale images of resolution 732 x 1142 px, which are later cropped to specifically display the manipulation workspace, marked off by the blue tape. To ease cloth perception, we also normalize the images based on a constant affine transformation to create greater contrast between the cloth and background. We further blur the image to remove noise, making it more closely resemble the smooth images seen in the Gym-Cloth simulator. The entire workspace setup is depicted in Fig 9.3.

Our physical experiments evaluate four repetitions of each policy, which for repeatability of results, utilize a set of four initial fabric configurations, depicted in Fig 9.2.

During physical experiments, we project pick points onto a color-segmented mask of the fabric, and project place points within the workspace. Similarly, when evaluating learned policies in the simulator, we project pick points onto a mask of the fabric; however, in simulation, episodes are terminated early if the fabric leaves the workspace.

9.2 Evaluation Metrics

Our experiments evaluate the following switching criteria introduced in Chapter 7 to determine when the BC policy should stop training and be deployed in real:

1. (Reward value) Simulation reward with a value-based stopping condition,

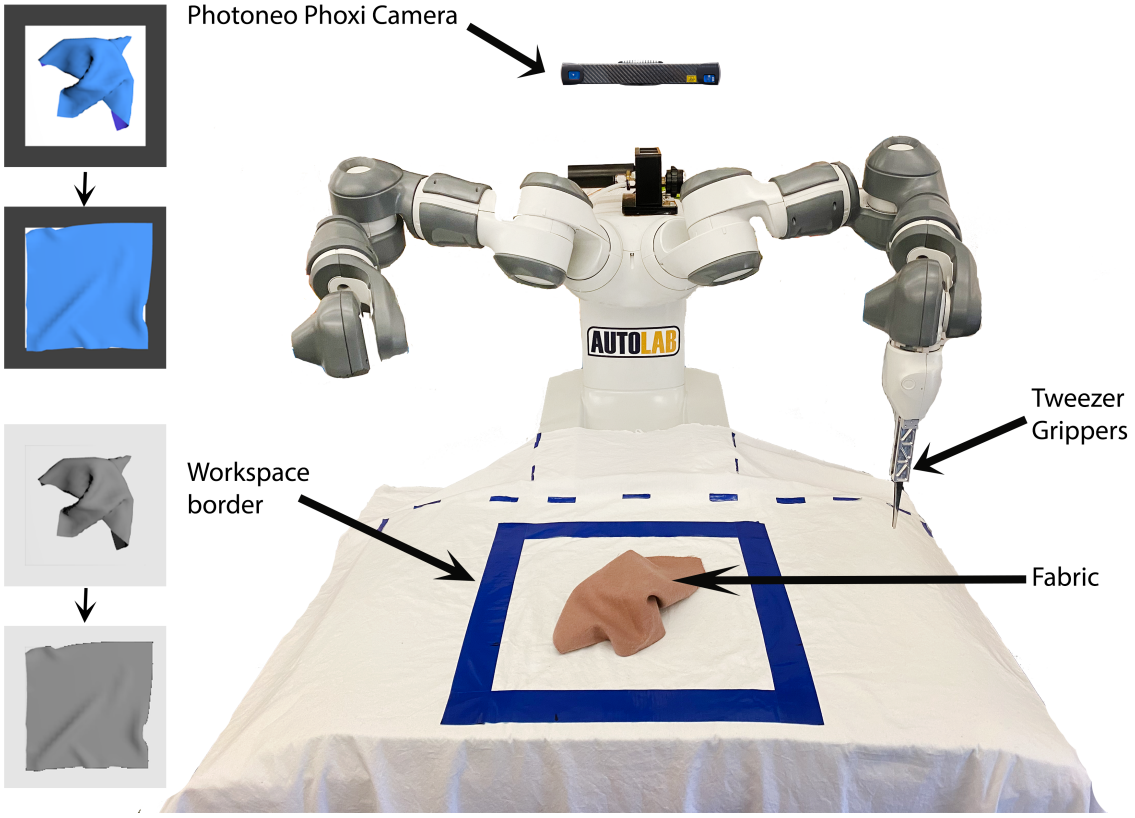


Figure 9.3: **Physical experiment and simulator setup.** We study sim-to-real switching in a fabric smoothing task, in an environment consisting of an ABB YuMi robot with a single tweezer gripper. An overhead Photoneo Phoxi Camera captures grayscale images. The manipulation workspace border is marked with blue tape, and the fabric is located within the workspace border. The physical workspace is designed to visually emulate the GymCloth simulator, as shown on the left; the top two lefthand images show example starting and ending configurations from an oracle smoothing policy, while the bottom two images show the same observations, processed to resemble the grayscale images taken by the Photoneo Phoxi Camera.

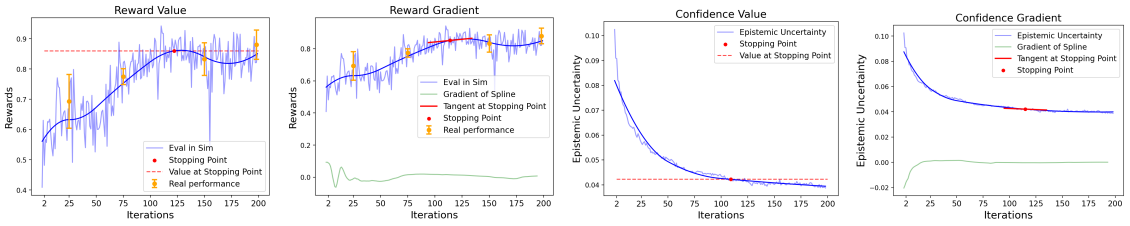


Figure 9.4: **Determining Stopping Points for Various Switching Criteria.** On all graphs, the dark blue curves are splines fit to the data to mitigate noise when approximating the gradient and determining if the value-based threshold has been met. **Left Two Graphs:** The simulation reward comes from evaluating the policy in the GymCloth simulation environment and determining the fabric coverage of the final configuration; curves are averaged over 5 episode rollouts in GymCloth. For comparison with real, the orange points correspond to mean performance in real of the BC policy selected at that iteration. The error bars correspond to the standard error across four runs. The stopping points (red point) are determined to be at iteration 124 for reward value and 116 for the reward gradient. **Right Two graphs**: The epistemic uncertainty is calculated at each iteration over a holdout set of 200 demonstration episodes and with five ensemble members. The confidence value determines the stopping point be 111, while the confidence gradient determines it to be 117.

2. (Reward gradient) Simulation reward with a gradient-based stopping condition,
3. (Confidence value) Epistemic uncertainty with a value-based stopping condition, and
4. (Confidence gradient) Epistemic uncertainty with a gradient-based stopping condition.

We evaluate the performance of these switching criteria via the following metrics, calculated during deployment in physical experiments after switching (averaged over 4 episodes in each case):

1. Final coverage
2. Improvement ratio (final coverage / initial coverage)
3. Number of actions per episode.

9.3 Results

The switching calculation is illustrated for each metric in Fig. 9.4, which displays the metrics and stopping point for each method. In each case, the stopping point is between 100 and 125 iterations out of a maximum of 200 learning iterations; the values are relatively close because all stopping condition parameters were tuned via cross-validation in simulation.

Table 9.1 summarizes the experimental results. We see the average final coverage achieved across all four stopping criteria is 88.05%, which is larger than the average initial coverage by a factor of 1.82. Thus, all four switching conditions are well-correlated with performance in real. We also plot fabric coverage over time in Figure 9.5, in which we see that 200 learning iterations—the maximum number considered—are not necessary to achieve competitive performance. Rather, our switching criteria identify an earlier stopping time that results in similar final coverage to the 200-iteration comparison, which does not allow early stopping.

Comparing the four stopping conditions, we see that the epistemic uncertainty metric yields higher final coverage than the simulation reward for both the value-based and gradient-based stopping conditions. Similarly, the value-based stopping condition showed higher performance than the gradient stopping condition for both the epistemic uncertainty and simulation reward metrics. However, because the stopping condition iterations are close together (see Table 9.1), it is possible that these discrepancies partially reflect noise in the behavior cloning training process, rather than an inherent difference between the switching criteria.

In terms of other pros and cons of the methods, in Figure 9.4, we see that the epistemic uncertainty exhibits less iteration-to-iteration noise than the simulation performance, while simulation performance is easier to tune and interpret and only requires training a single behavior cloning policy (rather than a policy ensemble). Meanwhile, the value-based stopping condition is easier to tune than the gradient-based method. However, because it looks at a time-based trend rather than individual values, we believe that the gradient-based stopping condition may yield more stable values and improved generalization compared to the value-based stopping condition. Testing generalizability of the stopping conditions is an interesting direction for future work.

Method (iters)	Imp. ratio	Final	Actions
Rew val (124)	1.925 ± 0.070	0.909 ± 0.030	8.50 ± 1.30
Rew grad (116)	1.611 ± 0.091	0.804 ± 0.065	8.75 ± 1.08
Conf val (111)	1.964 ± 0.070	0.937 ± 0.004	6.00 ± 1.27
Conf grad (117)	1.819 ± 0.103	0.872 ± 0.032	8.75 ± 1.08
Final (200)	1.858 ± 0.040	0.880 ± 0.048	7.50 ± 1.30

Table 9.1: **Physical Experiment Results:** We report the final fabric coverage and improvement ratio (final / initial coverage) achieved in physical fabric smoothing experiments by the various switching policies: reward value (rew val), reward gradient (rew grad), confidence value (conf val), and confidence gradient (conf grad). Results are mean ± standard error over 4 episodes in each case, and the initial coverage averaged 0.481 (± 0.023 standard error) over all cases reported in the table. We also report the average number of actions (where each trajectory terminates upon reaching 10 actions or 92% coverage, whichever occurs first).

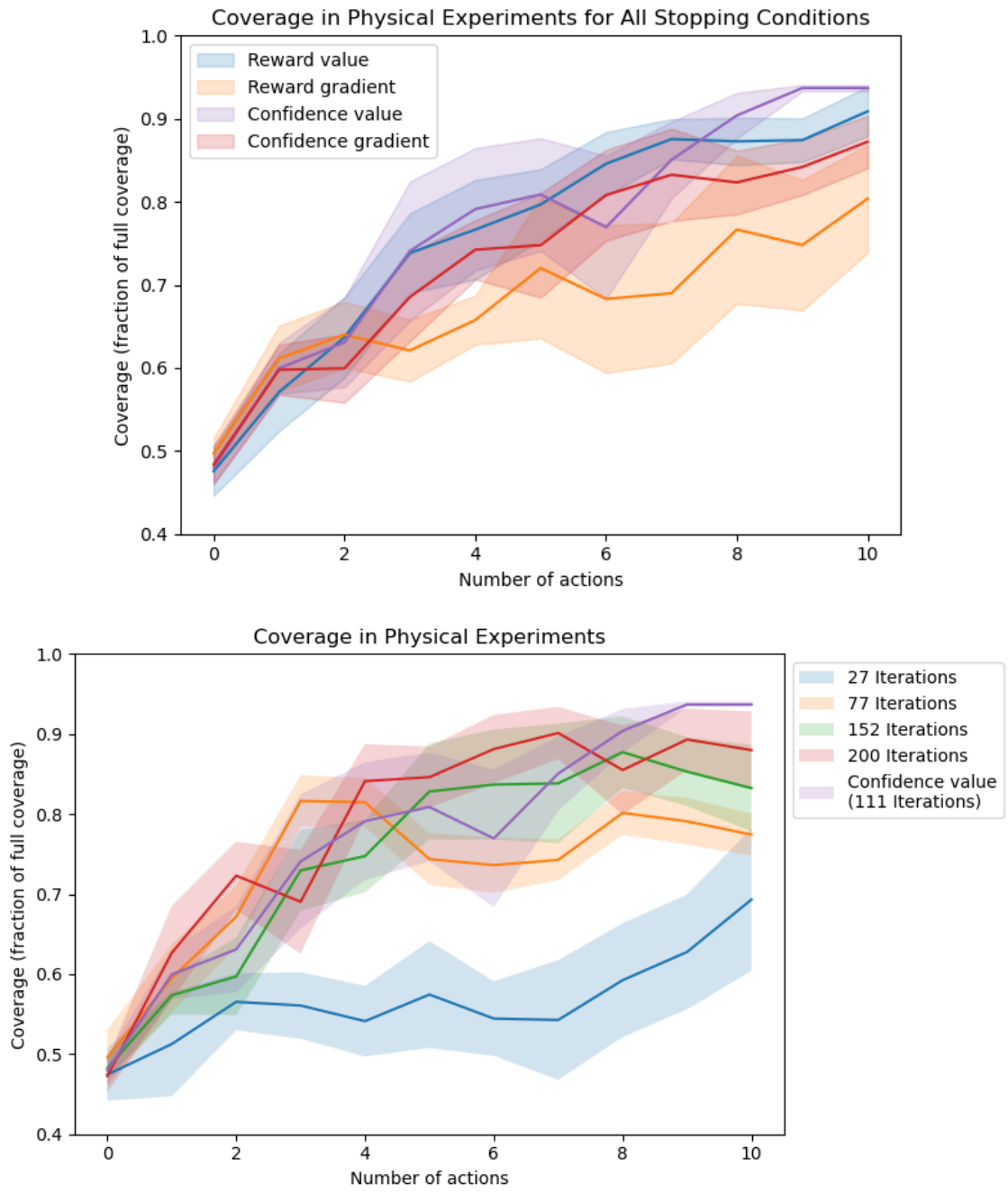


Figure 9.5: **Performance of learned policies in physical fabric smoothing experiments at various stopping points.** Left: Final physical fabric coverage achieved for each of the four stopping conditions. Right: Comparing the final physical fabric coverage for the confidence value stopping condition to various checkpoints. We see that the stopping conditions are largely competitive with 200 iterations (the maximum iteration number considered), but require significantly less training. Plots show mean $\pm$ standard err over 4 episodes. Note that to plot episodes that reach the target coverage of 92% in fewer than 10 actions, we repeat the final achieved coverage for the remainder of the 10-action budget.

Chapter 10

Limitations

10.1 Limitations with PG-BROIL and MPC-BROIL

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 [24], a continuously differentiable alternative to CVaR, but obtained similar results to CVaR. This would be a good limitation of PG-BROIL to better understand.

As an example of a limitation with PG-BROIL we show the same TrashBot environment as detailed in Section 5.2, where the agent's objective is to simultaneously collect pieces of trash while avoiding the gray region.

In the rightmost trajectory, we often found for extremely low values of λ that the agent would run off into a corner (as opposed to staying in the intended main region).

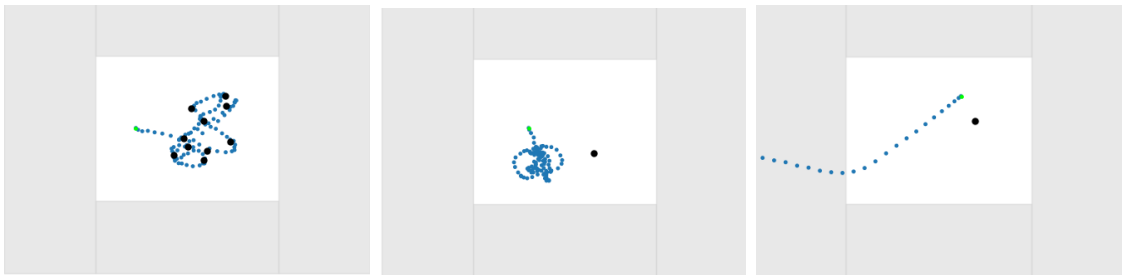


Figure 10.1: The left and middle images show trajectories for PG-BROIL with $\lambda = 0.8$ while the right image shows a failure case for $\lambda = 0.1$.

This is interesting behavior that we wish to understand the reasoning behind.

In the middle trajectory, we see the agent remaining in the white region but not collecting the piece of trash next to it. Although this is a relatively high value of $\lambda = 0.8$, we still see unintended behavior (this value of lambda is expected to additionally pick up trash as well as stay in the white region). This is also behavior we wish to better understand, especially since the leftmost trajectory shows this same λ value displaying intended behavior.

For MPC-BROIL, we also notice a small range of λ values that result in different behaviors. This is the same example as shown before in Section 7.1. We noticed that values ranging between $\lambda = 1$ and $\lambda = 0.93$ were almost identical to one another, having difficulties in seeing any differences in behavior between them. We also noticed that the trajectories between $\lambda = 0.92$ and $\lambda = 0$ only have very subtle differences. This could be because of the environment we used; perhaps another environment or task could better display a range of different behaviors for various values of λ.

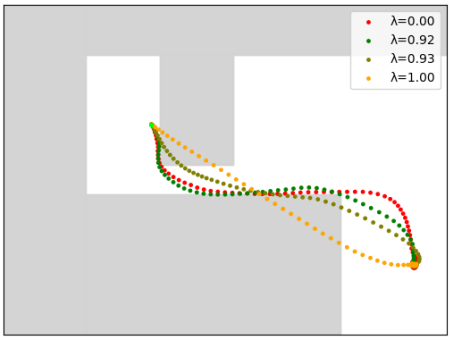


Figure 10.2: We evaluate MPC-BROIL on a Pointmass environment. An oracle dynamics model was used with a planning horizon of 20.

10.2 Limitations with Sim2Real Switching Criteria

The switching criteria which we evaluated were only tested for the case of behavior cloning. There are many other algorithms that are used for robotics and sim2real transfer, such as model-based planning and model-free reinforcement learning, and

even other imitation learning methods (such as GAIL). Therefore, our study on switching criteria isn't entirely inclusive as it only studies it for behavior cloning. It would be interesting to try to generalize and evaluate these criteria across several other types of algorithms, to test if they work just as well as they did in our behavior cloning example.

Chapter 11

Conclusions and Future Work

11.1 PG-BROIL and MPC-BROIL

We derived a policy gradient-style robust optimization approach, PG-BROIL, to allow for robust policy optimization in continuous MDPs. We also presented a sample-efficient, model predictive control approach, MPC-BROIL, which uses a learned dynamics model to optimize robust policies over distributions of reward functions. From both of these approaches, we saw that robust optimization can be scalable to large, continuous MDPs.

We tested on challenging domains, however one regret is having been unable to have ample time to test the algorithms on a real world robot. Real world domains are much more challening, and we believe that these algorithms can perform very well in real-world robotics contexts where there is uncertainty about the true objective. One example we thought about was navigating a robot arm around an uncertain region (perhaps of clutter) to a goal point. This would warrant the risk-averse behavior that our algorithms are capable of producing, since navigating directly through the clutter could result in damage to the robot arm. Bringing PG-BROIL and MPC-BROIL to the real world is an interesting project we would encourage future students to take on.

Future work also includes taking advantage of recent research on efficient non-linear Bayesian reward learning via Gaussian processes [7] and deep neural networks [12] to see if we can learn using more complex reward functions (as opposed to simple linear ones). It would be interesting to see how these rewards do compared to our linear ones. We also 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. Examining reasons for this instability and potential fixes would also be a

good direction for future work.

We also began work on learning rewards from latent features (in the case of MPC-BROIL). For this work, we got very close to getting it to work, however faced some issues. To learn the reward, we experimented with both Trajectory-ranked Reward EXtrapolation (T-REX) [11] and Bayesian Reward Extrapolation (B-REX) [10]. We were only able to get T-REX to properly work, however this gives us a single reward function as opposed to estimating a distribution, which we need for MPC-BROIL. This is also a great direction for future students to work on, since learning from the latent space would allow us to scale up to much more complex tasks (including robotic tasks).

Code for PG-BROIL can be found at `https://github.com/zaynahjaved/pg-broil`. Code for MPC-BROIL is not yet public since it is still underway.

11.2 Sim2Real Switching Criteria

We proposed and evaluated strategies for determining when to switch from training a learning algorithm in simulation to deploying it on a real physical system. We considered metrics based on performance in simulation performance and epistemic uncertainty. We tested two stopping conditions based on absolute thresholds and gradient tracking. We applied the switching criteria to a fabric smoothing task, where we studied sim2real transfer for behavior cloning. We found that the proposed switching criteria helped to save training time via early stopping. This can importantly save compute and energy resources and help to deploy policies onto robots in the real world more quickly.

While we focused on a single simulator and a single robot task, studying these switching criteria across multiple simulators and robot tasks would be useful to determine the generalizability of what we found. It could also be interesting future work to consider switching to real based on the learned policies' performance when transferring across simulators. Switching to real in other algorithmic contexts, outside of imitation learning, could also be studied, such as model-based planning and model-free reinforcement learning. It could also be interesting to observe two-way switching between learning in simulation and in physical experiments and whether this additionally accelerates the learning process.

Code for Sim2Real Switching Criteria can be found at `https://github.com/ernovoseller/BC_switching_criteria`.

Ending Notes

The 5th year master's program has been an invaluable experience that I highly recommend. I once again want to thank my advisor for giving me the chance to do this. I have learned a great deal about imiation learning as well as how to perform quality research. There are still some directions for future work on these projects that I would love to see or be a part of. I hope to continue research in the future, but for now I'm excited to begin a job at Meta!

Bibliography

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

[2] Joshua Achiam. "Spinning Up in Deep Reinforcement Learning". In: (2018). URL: https://spinningup.openai.com/.

[3] Joshua Achiam et al. "Constrained policy optimization". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 22–31.

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

[5] Philippe Artzner et al. "Coherent measures of risk". In: *Mathematical finance* 9.3 (1999), pp. 203–228.

[6] Marc G Bellemare et al. "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.

[7] Erdem Biyik et al. "Active Preference-Based Gaussian Process Regression for Reward Learning". In: *Proceedings of Robotics: Science and Systems (RSS)*. July 2020.

[8] Mariusz Bojarski et al. *End to End Learning for Self-Driving Cars*. 2016. DOI: 10.48550/ARXIV.1604.07316. URL: https://arxiv.org/abs/1604.07316.

[9] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[10] Daniel Brown, Scott Niekum, and Petrik Marek. "Bayesian Robust Optimization for Imitation Learning". In: *Neural Information Processing Systems (NeurIPS)*. 2020.

[11] Daniel Brown et al. "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 783–792.

[12] Daniel Brown et al. "Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences". In: *International Conference on Machine Learning*. 2020.

[13] Daniel S Brown and Scott Niekum. "Efficient probabilistic performance bounds for inverse reinforcement learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

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

[15] Paul Christiano et al. "Deep reinforcement learning from human preferences". In: *arXiv preprint arXiv:1706.03741* (2017).

[16] Kurtland Chua et al. *Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models*. 2018. arXiv: `1805.12114 [cs.LG]`.

[17] Erick Delage and Shie Mannor. "Percentile optimization for Markov decision processes with parameter uncertainty". In: *Operations research* 58.1 (2010), pp. 203–213.

[18] Freddy Delbaen. "Coherent risk measures on general probability spaces". In: *Advances in finance and stochastics*. Springer, 2002, pp. 1–37.

[19] Esther Derman et al. "Soft-robust actor-critic policy-gradient". In: *arXiv preprint arXiv:1803.04848* (2018).

[20] Yuqing Du et al. "Auto-Tuned Sim-to-Real Transfer". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), pp. 1290–1296.

[21] Benjamin Eysenbach et al. "Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers". In: *ICLR* (2021).

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

[23] Jaime F. Fisac et al. "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems". In: *IEEE Transactions on Automatic Control*. 2018.

[24] Hans Föllmer and Thomas Knispel. "Entropic risk measures: Coherence vs. convexity, model ambiguity and robust large deviations". In: *Stochastics and Dynamics* 11.02n03 (2011), pp. 333–351.

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

[26] Aditya Ganapathi et al. "Learning to Smooth and Fold Real Fabric Using Dense Object Descriptors Trained on Synthetic Color Images". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2020.

[27] Javier Garcıa and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.

[28] Alessandro Giusti et al. "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots". In: *IEEE Robotics and Automation Letters* (2016).

[29] Dylan Hadfield-Menell et al. "Inverse reward design". In: *Advances in neural information processing systems*. 2017, pp. 6765–6774.

[30] M. Heger. "Consideration of risk in reinforcement learning". In: *Machine Learning Proceedings*. 1994.

[31] Jonathan Ho and Stefan Ermon. "Generative Adversarial Imitation Learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 7461–7472.

[32] Sebastian Höfer et al. "Sim2Real in robotics and automation: Applications and challenges". In: *IEEE transactions on automation science and engineering* 18.2 (2021), pp. 398–400.

[33] Ryan Hoque et al. "LazyDAgger: Reducing Context Switching in Interactive Imitation Learning". In: *arXiv preprint arXiv:2104.00053* (2021).

[34] Ryan Hoque et al. "VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation". In: *Robotics: Science and Systems* (2020).

[35] Ryan* Hoque et al. "VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation". In: *Robotics Science and Systems*. 2020.

[36] Jessie Huang et al. "Learning safe policies with expert guidance". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9105–9114.

[37] Ahmed Hussein et al. "Imitation learning: A survey of learning methods". In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pp. 1–35.

[38] Stephen James et al. "RMA: Rapid Motor Adaptation for Legged Robots". In: *Conference on Computer Vision and Pattern Recognition (CVPR*. IEEE. 2019.

[39] Stephen James et al. "Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 12619–12629.

[40] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[41] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[42] Victoria Krakovna et al. "Specification gaming examples in AI". In: DeepMind Blog (2020).

[43] Aviral Kumar et al. "A Workflow for Offline Model-Free Robotic Reinforcement Learning". In: (2021).

[44] Jonathan Lacotte et al. "Risk-sensitive generative adversarial imitation learning". In: *arXiv preprint arXiv:1808.04468* (2018).

[45] Shuo Li and Osbert Bastani. "Robust Model Predictive Shielding for Safe Reinforcement Learning with Stochastic Dynamics". In: 2020.

[46] Zhongyu Li et al. "Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots". In: *International Conference on Robotics and Automation (ICRA)*. 2021.

[47] Vincent Lim et al. "Planar Robot Casting with Real2Sim2Real Self-Supervised Learning". In: (2021).

[48] Elita A Lobo, Mohammad Ghavamzadeh, and Marek Petrik. "Soft-Robust Algorithms for Handling Model Misspecification". In: *arXiv preprint arXiv:2011.14495* (2020).

[49] Jeffrey Mahler et al. "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics". In: *Robotics Science and Systems*. 2017.

[50] Maren Mahsereci et al. "Early Stopping without a Validation Set". In: *ArXiv* abs/1703.09580 (2017).

[51] Anirudha Majumdar et al. "Risk-sensitive Inverse Reinforcement Learning via Coherent Risk Models." In: *Robotics: Science and Systems*. 2017.

[52] Ajay Mandlekar et al. "What Matters in Learning from Offline Human Demonstrations for Robot Manipulation". In: (2021).

[53] D.Q. Mayne. "Model predictive control: Recent developments and future promise". In: *Automatica* 50 (Nov. 2014). DOI: 10.1016/j.automatica.2014.10.128.

[54] Bhairav Mehta et al. "Active Domain Randomization". In: *Conference on Robot Learning (CoRL)*. 2019.

[55] Fabio Muratore, Michael Gienger, and Jan Peters. "Assessing Transferability From Simulation to Reality for Reinforcement Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (2021), pp. 1172–1183.

[56] David Nass, Boris Belousov, and Jan Peters. "Entropic Risk Measure in Policy Search". In: *arXiv preprint arXiv:1906.09090* (2019).

[57] OpenAI et al. *Solving Rubik's Cube with a Robot Hand*. 2019. arXiv: 1910.07113 [cs.LG].

[58] Takayuki Osa et al. "An algorithmic perspective on imitation learning". In: *arXiv preprint arXiv:1811.06711* (2018).

[59] J Peters and S Schaal. "Reinforcement Learning of Motor Skills with Policy Gradients". In: *Neural Networks* 21.4 (2008), pp. 682–697.

[60] Dean A Pomerleau. "Efficient training of artificial neural networks for autonomous navigation". In: *Neural computation* 3.1 (1991), pp. 88–97.

[61] Dean A. Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 1. Morgan-Kaufmann, 1988. URL: https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

[62] Lutz Prechelt. "Automatic early stopping using cross validation: quantifying the criteria". In: *Neural Networks* 11.4 (1998), pp. 761–767. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(98)00010-0. URL: https://www.sciencedirect.com/science/article/pii/S0893608098000100.

[63] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley-Interscience, 2005.

[64] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning." In: *IJCAI*. Vol. 7. 2007, pp. 2586–2591.

[65] Fabio Tozeto Ramos, Rafael Possas, and Dieter Fox. "BayesSim: adaptive domain randomization via probabilistic inference for robotics simulators". In: (2019).

[66] E Ratner, D Hadfield-Mennell, and A Dragan. "Simplifying Reward Design through Divide-and-Conquer". In: *Robotics: Science and Systems*. 2018.

[67] Kevin Regan and Craig Boutilier. "Regret-based reward elicitation for Markov decision processes". In: *Conference on Uncertainty in Artificial Intelligence (UAI)*. 2009, pp. 444–451. ISBN: 978-0-9749039-5-8. arXiv: 1205.2619.

[68] R Tyrrell Rockafellar, Stanislav Uryasev, et al. "Optimization of conditional value-at-risk". In: *Journal of risk* 2 (2000), pp. 21–42.

[69] Stéphane Ross and Drew Bagnell. "Efficient reductions for imitation learning". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 661–668.

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

[71] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. "No-Regret Reductions for Imitation Learning and Structured Prediction". In: *CoRR* abs/1011.0686 (2010). arXiv: 1011.0686. URL: http://arxiv.org/abs/1011.0686.

[72] Reazul Hasan Russel, Bahram Behzadian, and Marek Petrik. "Entropic Risk Constrained Soft-Robust Policy Optimization". In: *arXiv preprint arXiv:2006.11679* (2020).

[73] Andrei A Rusu et al. "Sim-to-real robot learning from pixels with progressive nets". In: *Conference on Robot Learning*. PMLR. 2017, pp. 262–270.

[74] Dorsa Sadigh et al. "Active Preference-Based Learning of Reward Functions." In: *Robotics: Science and Systems*. 2017.

[75] Anirban Santara et al. "RAIL : Risk-Averse Imitation Learning Extended Abstract". In: *arXiv:1707.06658* (2018).

[76] John Schulman et al. "High-dimensional continuous control using generalized advantage estimation". In: *arXiv preprint arXiv:1506.02438* (2015).

[77] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[78] John Schulman et al. "Trust Region Policy Optimization". In: *arXiv preprint arXiv:1707.06347* (2017).

[79] Daniel Seita et al. "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 9651–9658.

[80] Yun Shen et al. "Risk-sensitive Reinforcement Learning". In: *Neural Computation*. Vol. 26. 2014.

[81] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[82] Umar Syed, Michael Bowling, and Robert E Schapire. "Apprenticeship learning using linear programming". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1032–1039.

[83] A. Tamar, Y. Glassner, and S. Mannor. "Policy Gradients Beyond Expectations: Conditional value-at-risk". In: *CoRR*. 2014.

[84] Aviv Tamar, Yonatan Glassner, and Shie Mannor. "Optimizing the CVaR via sampling". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[85] Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. "Worst Cases Policy Gradients". In: *Conf. on Robot Learning (CoRL)* (2019).

[86] Yichuan Charlie Tang, Jian Zhang, and Ruslan Salakhutdinov. "Worst Cases Policy Gradients". In: *Proceedings of the Conference on Robot Learning*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, 30 Oct–01 Nov 2020, pp. 1078–1093. URL: http://proceedings.mlr.press/v100/tang20a.html.

[87] Ajay Tanwani. "DIRL: Domain-Invariant Representation Learning for Sim-to-Real Transfer". In: *Conf. on Robot Learning (CoRL)*. IEEE. 2020.

[88] Ajay Kumar Tanwani. "DIRL: Domain-Invariant Representation Learning for Sim-to-Real Transfer". In: *Conference on Robot Learning (CoRL)*. 2020.

[89] Yuval Tassa et al. *dm_control: Software and Tasks for Continuous Control*. 2020. arXiv: 2006.12983 [cs.RO].

[90] Brijen Thananjeyan et al. "ABC-LMPC: Safe Sample-Based Learning MPC for Stochastic Nonlinear Dynamical Systems with Adjustable Boundary Conditions". In: *Workshop on the Algorithmic Foundations of Robotics*. 2020.

[91] Brijen Thananjeyan et al. "Recovery RL: Safe Reinforcement Learning with Learned Recovery Zones". In: *Robotics and Automation Letters (RA-L)*. IEEE. 2021.

[92] Brijen Thananjeyan et al. "Safety Augmented Value Estimation from Demonstrations (SAVED): Safe Deep Model-Based RL for Sparse Cost Robotic Tasks". In: *Robotics and Automation Letters (RAL)* (2020).

[93] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[94] Faraz Torabi, Garrett Warnell, and Peter Stone. "Behavioral cloning from observation". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 4950–4957.

[95] Eugene Valassakis, Zihan Ding, and Edward Johns. "Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5372–5379.

[96] Kim P. Wabersich, Raamadaas Krishnadas, and Melanie N. Zeilinger. "A Soft Constrained MPC Formulation Enabling Learning From Trajectories With Constraint Violations". In: *IEEE Control Systems Letters* 6 (2022), pp. 980–985. DOI: 10.1109/LCSYS.2021.3087968.

[97] Min Wen and Ufuk Topcu. "Constrained Cross-Entropy Method for Safe Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/34ffeb359a192eb8174b6854643cc046-Paper.pdf.

[98] Kelvin Xu et al. "Learning a prior over intent via meta-inverse reinforcement learning". In: *International Conference on Machine Learning* (2019).

[99] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. "On Early Stopping in Gradient Descent Learning". In: *Constructive Approximation* 26.2 (Aug. 2007), pp. 289–315. ISSN: 1432-0940.

[100] Jiakai Zhang and Kyunghyun Cho. "Query-efficient imitation learning for end-to-end autonomous driving". In: *arXiv preprint arXiv:1605.06450* (2016).

[101] Shangtong Zhang, Bo Liu, and Shimon Whiteson. "Mean-Variance Policy Iteration for Risk-Averse Reinforcement Learning". In: *Conference on Artificial Intelligence (AAAI)*. 2021.

[102] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2020, pp. 737–744.

[103] Brian D Ziebart et al. "Maximum entropy inverse reinforcement learning." In: *Aaai*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.