Towards Trustworthy Machine Learning

Adam Gleave

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-260
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-260.html

December 5, 2022

Copyright © 2022, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Towards Trustworthy Machine Learning

by

Adam Gleave

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Stuart Russell, Chair
Associate Professor Sergey Levine
Associate Professor Anca Dragan
Assistant Professor Chelsea Finn

Fall 2022

Towards Trustworthy Machine Learning

Copyright 2022
by
Adam Gleave

Abstract

Towards Trustworthy Machine Learning

by

Adam Gleave

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Stuart Russell, Chair

Real-world applications of machine learning often have complex objectives and safety-critical constraints. Contemporary machine learning systems excel at achieving high average-case performance at tasks with simple procedurally specified objectives, but they struggle at many more demanding real-world tasks. In this thesis, we work towards developing *trustworthy machine learning* systems that understand human values and reliably optimize them.

Machine learning's key insight was that it is often easier to *learn* an algorithm than to write it down directly—yet many machine learning systems still have a hard-coded, procedurally specified objective. The field of reward learning applies this insight to instead learn the objective itself. As there is a many-to-one mapping between reward functions and objectives, we start by introducing the notion of *equivalence classes* consisting of reward functions that specify the same objective.

In the first part of the dissertation, we apply this notion of equivalence classes to three distinct settings. First, we study reward function identifiability: what set of reward functions is compatible with the data? We start by categorizing the equivalence classes of reward functions that induce the same data. By comparing these to the aforementioned optimal policy equivalence class, we can determine whether a given data source provides sufficient information to recover the optimal policy.

Second, we address the fundamental question of how similar or dissimilar two reward function equivalence classes are. We introduce a distance metric over these equivalence classes, the *Equivalent-Policy Invariant Comparison (EPIC)*, and show rewards with low EPIC distance induce policies with similar returns even under different transition dynamics. Finally, we introduce an interpretability method for reward function equivalence classes. The method selects the easiest to understand representative from the equivalence class, and then visualizes the representative function.

In the second part of the dissertation, we study the adversarial robustness of models. We

start by introducing a physically realistic threat model consisting of an *adversarial policy* acting in a multi-agent environment so as to create *natural* observations that are adversarial to the defender. We train the adversary using deep RL against a frozen state-of-the-art defender that was trained via self-play to be robust to opponents. We find this attack reliably wins against state-of-the-art simulated robotics RL agents, and superhuman Go programs.

Finally, we investigate ways to improve agent robustness. We find adversarial training is ineffective, however population-based training offers hope as a partial defense: it does not prevent the attack, but it does increase the computational burden of the attacker. Using explicit planning also helps, as we find that defenders with large amounts of search are harder to exploit.

For humanity, that we may overcome our challenges and achieve a flourishing future.

Contents

List of Figures

List of Tables

Acknowledgments

My journey into research started in distributed systems, with the encouragement and advice of Malte Schwarzkopf and Ionel Gog then at the University of Cambridge. Although my thesis has turned out to be on an entirely unrelated topic, I continue to cherish the research skills I gained from working with both of them, especially advice on clear technical writing. I was fortunate to then work with Christian Steinruecken and Zoubin Ghahramani during my master's degree on probabilistic models for text. I am grateful for them introducing me to machine learning research and teaching me to be experimentally rigorous.

Despite thoroughly enjoying these research experiences, I almost ended up not pursuing research at all. Indeed, I spent a year as a quantitative trader prior to starting my PhD. I am indebted to Beth Barnes for overcoming my inertia and convincing me to take AI safety research seriously as a career option.

I was fortunate to be advised by Stuart Russell during my PhD. Looking back, I am particularly grateful to have been given an exceptional amount of freedom. I spent most of my first year reading widely and discussing different research agendas with my peers. At the time, I could not shake the feeling I was wasting my time, as I saw many of my peers with concrete research projects well on the way to publication. However, I am now glad I spent my time reflecting in this way, as it set me up for success later on in the PhD. So, thank you for encouraging me to be ambitious and do good research, regardless of what conference reviewers may think of it!

I am also grateful for the advice of Sergey Levine in my work on adversarial policies, by far my best-known work. He has always been able to spot the heart of the technical problem, and I am frankly always astounded by how much material we can cover in just a half hour meeting! Thank you also for your advice on framing and communicating research findings.

I have also learned an incredible amount from the people I have worked with at Berkeley and elsewhere. Thanks in particular to Rohin Shah, Daniel Filan, and Dylan Hadfield-Menell for helping me develop my views on AI safety. I am also grateful for Lawrence Chan, Paul Christiano, Sam Toyer, Cassidy Laidlaw, Rachel Freedman, Scott Emmons, Alex Turner, Daniel Kokotajlo, Micah Carroll, Richard Ngo, and Ajeya Cotra for enlightening conversations on this topic. Of course I am also grateful to my many collaborators, including Tony Wang, Michael Dennis, Nora Belrose, Tom Tseng, Antonin Raffin, Ashley Hill, Juan Rocamonde, Maximilian Ernestus, Sören Mindermann, Cody Wild, and Steven Wang.

I am also fortunate to have had the opportunity to mentor a number of exceptional people. Although nominally I was meant to be teaching them, I have no doubt learned a tremendous amount as both a researcher and an advisor from working with them all. So thanks to Leo Richter, Pavel Czempin, Erik Jenner, Joar Skalse, Matthew Farrugia-Roberts, Lauro Langosco, Oliver Richardson, Eric Michaud, Sergia Volodin, Pedro Freire, Neel Kant, Aaron Tucker, Yawen Duan, and Lev McKinney for working with me!

Outside of Berkeley, I was lucky to be advised by Jan Leike and Geoffrey Irving during successive internships at DeepMind. Thanks in particular to Jan for encouraging me to continue to pursue reward function evaluation even after a series of disappointing results. I am glad we persisted, as the final EPIC metric was both theoretically pleasing and practically relevant. Working with Geoffrey was my introduction to the fascinating area of language model alignment research. I am particularly grateful for his efforts to make me and others on the team feel welcome even in the midst of a remote start in a pandemic.

Throughout my PhD I have benefited from support from a variety of staff members. Thanks to staff from CHAI, BAIR, EECS, and ERSO for a variety of operational support, in particular Martin Fukui, Angie Abbatecola, Jean Nguyen, and Judy Tam. I have also been fortunate to have spent time working from the Lightcone and Constellation co-working spaces—my thanks to their respective teams for providing a productive workplace environment and enabling many stimulating conversations with other researchers. I am also grateful to Euan McLean and Wes Cowley for editing this manuscript, and to Alyse Spiehler for developing several illustrations.

My PhD would have been much harder without the support and encouragement of many friends. It would be difficult to list them all, but I would like to specifically thank Maja Cernja, Vaidehi Agarwalla, Michael Hsu, Rick Korzekwa, Andy Jones, Daniel Ziegler, George Howes, Ethan Perez, Linchuan Zhang, Oliver Habryka, and Erin Grant. Last but by no means least, thank you to my family for your support throughout graduate school.

This dissertation includes results developed in collaboration with my aforementioned advisors and Tony Wang, Nora Belrose, Tom Tseng, Yawen Duan, Viktor Pogrebniak, Joseph Miller, Pavel Czempin, Erik Jenner, Joar Skalse, Matthew Farrugia-Roberts, Michael Dennis, Cody Wild, and Neel Kant. Portions of this text have been adapted from Skalse et al. [155], Gleave et al. [55, 54], Jenner and Gleave [73], Wang et al. [171], and Czempin and Gleave [37].

Chapter 1

Introduction

1.1 A decomposition of trustworthy machine learning

The field of machine learning (ML) has made remarkable progress towards building automated systems that achieve high average performance on procedurally specified objectives. In reinforcement learning, we have seen superhuman performance in a variety of board games [151, 152, 161, 175] and real-time strategy games [21, 169]. Meanwhile, many image classifiers have reached or surpassed human performance [126, 136, 148, 125], and language models outperform humans on next-token prediction [150].

An outside observer might expect these advances to allow machine learning systems to automate a wide variety of tasks previously performed by humans. However, real-world tasks have so far proved much more difficult than benchmark tasks that *prima facie* appear more complex. Even a task as simple as peg insertion from pixels has a non-trivial reward function that must usually be learned [166, IV.A]. Tasks involving human interaction can have far more complex reward functions that users may not even be able to introspect on. We believe this disconnect demonstrates the need for progress on *trustworthy machine learning*.

We propose two necessary conditions for trustworthy machine learning in the context of sequential decision-making agents. First, the *agent objective* should be aligned with the human principal, otherwise optimizing it will clearly result in undesired consequences. Second, the agent should be *robust*, reliably pursuing this objective across a wide range of scenarios.

Correctly specifying the agent objective can be challenging. Procedurally specified objectives usually miss important frame conditions [61]. Instead of procedurally specifying the objective, a designer may build a system that infers the objective from human data [74]. Learned rewards often provide a higher-fidelity representation of the designer's objective than procedural objectives. For example, Stiennon et al. [156] find that optimizing a learned reward model produces summaries that human evaluators prefer over those produced by a model optimizing the widely used ROUGE metric.

However, there remains a large gap between machine learning systems ability to learn our objectives, versus optimizing objectives for tasks that can be easily specified. While superhuman performance was achieved several years ago in the strategically complex video game StarCraft [169], there is still no extant AI system that can perform strategically simpler but hard-to-specify tasks in Minecraft such as “build a beautiful waterfall and take a scenic picture of it” [147].

Critically, it is not enough for the optimization objective to be highly *correlated* with the designer’s intent. Optimization processes are excellent at finding edge cases in objectives, often finding the cases where this correlation breaks down, an instance of Goodhart’s law [59, 101]. Indeed, prior work has shown that both procedurally specified [118] and learned reward functions [104] are vulnerable to such “reward hacking”. Consequently, it is necessary to hold learned objectives to a higher standard than we typically do for machine learning models. It is important learned reward objectives accurately represent human preferences, not a shallow facsimile of them, such as a model built on top of spurious correlates.

Even if the objective is correctly specified and robust, it is common for the *policy* produced by the optimization process to be lacking in robustness. That is, the policy may not be able to achieve its objective across a wide range of situations. An illustrative example is autonomous driving. The RALPH autonomous vehicle completed a 3000-mile trip across the United States with 96% autonomous steering in 1995 [128]. Nearly three decades later, firms have only just begun limited deployments of commercial autonomous vehicles [81]. Obtaining good performance in the average case was achievable on a shoestring budget several decades ago, but reliably handling the long-tail of real-world scenarios is only just achievable now after billions of dollars of investment.

Perhaps surprisingly, we find that in many cases there is no framework to even rigorously *evaluate* whether a given machine learning system is trustworthy. Our key contribution is to formalize the notion of trustworthiness in several important settings, providing theoretical and empirical understanding of the limits and potential of different system designs. Concretely, for agent objectives we establish an upper bound of performance for idealized reward learning algorithms, develop a distance metric over reward functions which bounds the regret of resulting optimal policies, and a novel approach for explainable reward models. For agent robustness, we introduce a novel and physically realistic threat model, demonstrate that state-of-the-art and even superhuman systems are vulnerable to our attack, and propose several defenses.

1.2 Agent objectives: trying to do the right thing

A key insight behind machine learning is that it is often simpler to specify an objective for a task than an algorithm to procedurally solve the task. Designing systems to optimize objectives such as reward functions or loss functions has allowed systems to “learn” how to perform complex tasks such as image classification and game playing. These learned systems often outperform more classical artificial intelligence approaches that rely on hard-coded algorithms or features. Unfortunately, the reward functions for most real-world tasks are difficult or impossible to specify procedurally. This suggests that we may benefit from taking learning a step further: replacing the hard-coded objective with one that is learned from human feedback.

There exist a variety of methods to learn reward functions. Inverse reinforcement learning (IRL) [138, 112, 131, 183, 47, 49, 11] is a common approach that works by inferring a reward function from demonstrations. An alternative approach is to learn from *preference comparisons* between two trajectories [4, 173, 33, 141, 68, 184]. T-REX [27] is a hybrid approach, learning from a *ranked* set of demonstrations. More directly, Cabi et al. [29] learn from "sketches" of cumulative reward over an episode.

Reward learning has achieved some notable successes. For example, it has achieved state-of-the-art performance in developing language models that follow instructions and are helpful assistants [116, 12]. However, the learned reward is often fragile. For example, Stiennon et al. [156, Table 29] find that optimizing the learned reward leads to gibberish summaries of articles that achieves a high learned reward but do not solve the task at hand. They work around this by applying a Kullback–Leibler (KL) penalty to keep the learned policy close to a base imitation-learned model, an approach that is followed in later work [116, 12]. In other words, better performance is obtained when only *weakly* optimizing the learned reward—actually maximizing it leads to *worse* performance.

This suggests a serious deficiency in current reward learning approaches. A natural question to ask is to what degree we are limited by current *algorithms*, versus the *human data* collected. In Chapter 3, we examine the degree to which reward functions can be identified from a given data source, developing a hierarchy of informativeness of different data sources. These results provide an upper bound on reward learning performance.

Notably we find there is significant unidentifiability for many data sources. That is, no matter how much data are collected or how sophisticated the algorithm, there remains irreducible ambiguity when learning from widely used sources such as demonstrations. In some cases this ambiguity may even impact the optimal policy, causing a reward learning agent to take worse actions than if it had received data with less ambiguity.

It is natural to judge reward learning algorithms based on how close they are to attaining this infinite-data upper bound on a given amount of data. However, prior work in reward learning has largely not evaluated the learned reward directly. Instead, it has evaluated the learned reward functions by first training a policy by applying RL on the learned reward. This policy is then evaluated, whether by measuring the ground-truth reward attained or by direct human inspection.

Unfortunately, this policy evaluation method cannot distinguish between the learned reward function failing to reflect user preferences, and the RL algorithm failing to correctly optimize the learned reward. Moreover, the policy evaluation method can only tell us about behavior in the evaluation environment, but the reward may incentivize very different behavior in even a slightly different deployment environment.

In Chapter 4, we introduce an alternative approach that can directly evaluate the learned reward function. Our *Equivalent-Policy Invariant Comparison (EPIC)* distance quantifies the difference between two reward functions directly, without a policy optimization step. Importantly, EPIC is invariant to positive affine transformations and potential shaping [111] of reward functions, which are both guaranteed to never change which policy is optimal. Furthermore, we find in Chapter 5 that EPIC distance bounds the regret of optimal policies even under different transition dynamics. We furthermore confirm this result holds empirically even when the assumptions in the theorem are violated.

EPIC is of immediate applicability for benchmarking reward learning algorithms, as it can be used to compare the learned reward function to a "ground-truth" reward function provided in the benchmark. However, in practical settings of reward learning there is no ground-truth reward! In this setting, EPIC can be used to cluster together similar learned reward functions that were perhaps learned by different algorithms or from different data sources, allowing us to analyze just a single representative of each cluster. But an additional technique is required to perform this analysis.

In Chapter 6, we introduce methods for *interpreting* learned reward functions. We leverage the same key insight from EPIC: we do not care about the reward function per se, but rather the *equivalence class* of rewards that always induce the same optimal policy. Consequently, we wish to choose a *representative* of this equivalence class that is easiest to understand. We introduce a method to automatically search for interpretable representatives, and we demonstrate this approach with several commonly used visualization techniques.

1.3 Agent robustness: achieving high levels of reliability

Machine learning excels in settings where data is independent and identically distributed (iid), and where achieving high average-case performance is sufficient. However, the resulting models are often fragile. Adversarial examples show that carefully chosen perturbations that are barely perceptible to a human can trigger errors in otherwise highly capable models [158]. Even when the iid assumption holds, safety-critical applications often require very high levels of reliability which contemporary methods often struggle to reach on complex, real-world datasets [64].

This suggests a need to focus not just on training agents to have good average-case performance, but also on them being *robust* in the face of rare and possibly adversarially chosen inputs. Prior work has shown that, similar to image classifiers, policies trained by reinforcement learning in video game environments are vulnerable to image perturbations [67]. However, an attacker cannot usually directly modify another agent's observations. This might lead one to wonder: is it possible to attack an RL agent simply by choosing an *adversarial policy* acting in a multi-agent environment so as to create *natural* observations that are adversarial?

In Chapter 7, we formalize this threat model and demonstrate the existence of adversarial policies in zero-sum games between simulated humanoid robots with proprioceptive observations. We train the adversary using deep RL against a fixed state-of-the-art "defender" policy that was trained via self-play to be robust to opponents. The adversarial policies reliably win against the defenders but generate seemingly random and uncoordinated behavior. We find that these policies induce substantially different activations in the defender policy network than when the defender plays against a normal opponent. Videos are available at `https://adversarialpolicies.github.io/`.

Although the simulated robotics agents that we attacked were state-of-the-art, due to the challenging nature of robotics they were still well below human performance. This raises the question: are adversarial policies a vulnerability of self-play policies *in general*, or simply an artifact of sub-human policies? In Chapter 9, we perform a similar attack against the state-of-the-art Go-playing AI system, KataGo. Our attack achieves a >99% win rate against KataGo without Monte Carlo tree search (MCTS) and a >80% win rate when KataGo uses enough search to be near-superhuman.

In Chapter 10, we investigate ways to improve agent robustness. We find that training a defender against a specific adversary protects against that adversary, but repeating the attack method finds a new adversarial policy that can exploit the hardened defender. However, we find that population-based training of the defender can increase the amount of training time the attack requires to find an exploit. Additionally, we find that increasing the search depth and defenses using domain-specific knowledge of the environment can improve robustness.

1.4 Overview

The remainder of the dissertation is organized as follows:

- Chapter 2 introduces background material used throughout the work, including on sequential decision-making such as Markov Decision Processes and the Bellman backup. We also define an equivalence class over reward functions consisting of reward functions guaranteed to have the same optimal policy.

- In Chapter 3, we start our study of learning agent objectives by studying the information content of different data sources used for reward learning. This gives fundamental upper bounds on the performance of any algorithm learning from that data source, no matter how many samples it sees. More generally, the framework introduced allows us to study what kinds of ambiguity certain uses of reward functions, such as policy optimization or policy comparison, are sensitive to.

- Chapter 4 introduces the EPIC distance over reward functions, which is invariant on the equivalence class defined in Chapter 2. The EPIC distance allows us to directly compare reward functions, rather than the policies they induce. Reward learning algorithms can be benchmarked by comparing their learned reward to a ground-truth reward. In real-world applications where a ground-truth reward is absent, EPIC can be used to cluster similar reward functions together.

- Chapter 5 shows that the EPIC distance between two rewards bounds the difference in return between their respective optimal policies. This result holds under arbitrary transition dynamics with only minor assumptions. Moreover, we find empirically that the EPIC distance of a reward from the ground-truth predicts the ground-truth performance of policies optimized using that reward, even when the assumptions made in the theorem are relaxed.

- Chapter 6 introduces a method to explain learned reward functions. In particular, it chooses the easiest-to-interpret representative of the equivalence class over reward functions, and then visualizes it using existing methods.

- In Chapter 7, we focus on the other half of trustworthy machine learning: agent robustness. We introduce a novel and physically realistic threat model, *adversarial policies*, where an attacker acts in an environment shared with the defender. The game is assumed to be zero-sum, and the defender agent was trained to win this game, so should in principle already be robust to attack.

- Chapter 8 applies this threat model to games between simulated humanoid robots. We find adversarial policies against state-of-the-art agents. Our attack is simple: we freeze the defender's network, then train the adversary on the resulting single-agent RL problem. This simple black-box attack nonetheless finds highly successful adversarial policies with a fraction of the compute used to train the defender. Moreover, it wins not by playing the intended game, but by performing seemingly random motion that confuses the defender.

- Chapter 9 describes an exploit of professional-level Go-playing AI system, KataGo, demonstrating that advances in average-case capabilities are not sufficient to provide adversarial or worst-case guarantees. Similar to Chapter 8, we find that the resulting adversarial policies win in surprising ways by fooling the defender.

- Chapter 10 investigates population-based RL as a defense against adversarial policies. We find that it improves robustness, as measured by the amount of training time needed by an adversary to exploit the defender. However, it does so at a significant increase in computational cost, making it most useful when the defender has more computational resources than the attacker.

- In Chapter 11, we summarize our findings and describe avenues for future work.

Chapter 2

Preliminaries

2.1 Markov Decision Processes

We formalize the sequential decision-making problem as a *Markov Decision Process* [MDP; 157, §3].

Definition 2.1.1. A *Markov Decision Process (MDP)* $M = (\mathcal{S}, \mathcal{A}, \gamma, \mu_0, \tau, R)$ consists of a set of states $\mathcal{S}$ and a set of actions $\mathcal{A}$, a discount factor $\gamma \in [0, 1]$, an initial state distribution $\mu_0(s)$, a transition distribution $\tau(s' \mid s, a)$ specifying the probability of transitioning to s' from s after taking action a, and a reward function $R(s, a, s')$ specifying the reward upon taking action a in state s and transitioning to state s'.

In this dissertation, unless otherwise stated we assume a discounted ($\gamma < 1$) infinite-horizon MDP. Our results can usually be generalized to undiscounted ($\gamma = 1$) MDPs subject to regularity conditions needed for convergence.

In the case of reward learning, we do not have access to the ground-truth reward R, but typically do still have access to an environment with which we can interact. This environment can be modeled as an MDP without a reward function, often called an MDP\R, $M^- = (\mathcal{S}, \mathcal{A}, \gamma, \mu_0, \tau)$. Typically, only the state space $\mathcal{S}$, action space $\mathcal{A}$, and discount factor γ are known exactly, with the initial state distribution μ_0 and transition dynamics τ only observable from interacting with the environment M^-.

We distinguish states in the support of μ_0 as *initial states* and states s with $\tau(s|s, a) = 1$ and $R(s, a, s) = 0$ for all a as *terminal states.*

We represent the *transition* from state s to state s' using action a as the tuple $x = (s, a, s')$. A *trajectory* is an infinite sequence of concatenated transitions $\xi = (s_0, a_0, s_1, a_1, s_2, \ldots)$, and a *trajectory fragment* of length n is a finite sequence of n concatenated transitions $\zeta = (s_0, a_0, s_1, \ldots, a_{n-1}, s_n)$.

We define the *return function* G over trajectory (fragments) as the cumulative discounted reward

$$G(\zeta) = \sum_{t=0}^{n-1} \gamma^t R(s_t, a_t, s_{t+1})$$

for a trajectory fragment ζ of length n, and for trajectories as

$$G(\xi) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}).$$

A *stochastic policy* $\pi(a \mid s)$ assigns probabilities to taking each action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. Together with an MDP's transition distribution τ, a policy π induces a distribution of trajectories starting from each state. We denote such a trajectory starting from s with the random variable Ξ_s, and its remaining components with random variables A_0, S_1, A_1, S_2, and so on.

Given an MDP and a policy π, the *value function* encodes the expected return from a state, $V_\pi(s) = \mathbb{E}_{\Xi_s \sim \pi, \tau}\left[G(\Xi_s)\right]$, and the *Q-function* of π encodes the expected return given an initial action, $Q_\pi(s,a) = \mathbb{E}_{\Xi_s \sim \pi, \tau}\left[G(\Xi_s) \mid A_0 = a\right]$. Q_π and V_π each (uniquely) satisfy a *Bellman equation*:

$$Q_\pi(s,a) = \mathbb{E}_{S' \sim \tau(s,a)}\left[R(s,a,S') + \gamma V_\pi(S')\right], \quad V_\pi(s) = \mathbb{E}_{A \sim \pi(s)}\left[Q_\pi(s,A)\right] \tag{2.1}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Their difference, $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$, is the *advantage function.*

We further define a *policy evaluation function*, $\mathcal{J}$, encoding the expected return from following a particular policy in an MDP, $\mathcal{J}(\pi) = \mathbb{E}_{S_0 \sim \mu_0}\left[V_\pi(S_0)\right]$. $\mathcal{J}$ induces an order over policies. A policy maximizing $\mathcal{J}$ is an *optimal policy*, denoted $\pi_\star$. Similarly, $Q_\star$, $V_\star$, and $A_\star$ denote the Q-, value, and advantage functions of an optimal policy respectively. Since $\mathcal{J}$ may have multiple maxima, we often discuss the *set* of optimal policies. However, $Q_\star$, $V_\star$, and $A_\star$ are each unique.

2.2 Optimal-policy-preserving reward transformations

There are two main classes of reward transformations that never change the optimal policy: potential shaping and positive affine transformations. We introduce both of them below. Finally, we combine them into an equivalence class.

Potential shaping is a reward transformation that moves reward earlier or later in a trajectory [111]. It does not change the overall return G (up to a constant), so it leaves the optimal policy $\pi_\star$ unchanged.

Definition 2.2.1. Let $\gamma \in [0,1]$ be the discount factor, and $\Phi : \mathcal{S} \to \mathbb{R}$ a real-valued function. Then $R(s,a,s') = \gamma\Phi(s') - \Phi(s)$ is a *potential-shaping* reward, with *potential* Φ [111].

The expected return of potential shaping $\gamma\Phi(s') - \Phi(s)$ on a trajectory segment $(s_0, \cdots, s_T)$ is $\gamma^T \Phi(s_T) - \Phi(s_0)$. The first term $\gamma^T \Phi(s_T) \to 0$ as $T \to \infty$, while the second term $\Phi(s_0)$ only depends on the initial state. Thus, potential shaping does not change the set of optimal policies. Moreover, any additive transformation that is not potential shaping will, for some reward R and transition distribution τ, produce a set of optimal policies that is disjoint from the original [111].

The set of optimal policies is invariant to constant shifts $c \in \mathbb{R}$ in the reward; however, this can already be obtained by shifting Φ by $\frac{c}{\gamma - 1}$.* Scaling a reward function by a positive factor $\lambda > 0$ scales the expected return of all trajectories by λ, also leaving the set of optimal policies unchanged.

Below, we describe an equivalence class whose members are guaranteed to have the same optimal policy set in *any* MDP\R M^- with fixed $\mathcal{S}$, $\mathcal{A}$, and γ (allowing the unknown τ and μ_0 to take arbitrary values). This combines potential shaping and positive affine transformations.

Definition 2.2.2 (Reward Equivalence)**.** We define two bounded reward functions R_A and R_B to be *equivalent*, $R_A \equiv R_B$, for a fixed $(\mathcal{S}, \mathcal{A}, \gamma)$ if and only if there exists a constant $\lambda > 0$ and a bounded potential function $\Phi : \mathcal{S} \to \mathbb{R}$ such that for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$:

$$R_B(s, a, s') = \lambda R_A(s, a, s') + \gamma\Phi(s') - \Phi(s). \tag{2.2}$$

Proposition 2.2.3. *The binary relation $\equiv$ is an equivalence relation. Let $R_A, R_B, R_C : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be bounded reward functions. Then $\equiv$ is reflexive, $R_A \equiv R_A$; symmetric, $R_A \equiv R_B$ implies $R_B \equiv R_A$; and transitive, $(R_A \equiv R_B) \wedge (R_B \equiv R_C)$ implies $R_A \equiv R_C$.*

Proof. $R_A \equiv R_A$ since choosing $\lambda = 1 > 0$ and $\Phi(s) = 0$, a bounded potential function, we have $R_A(s, a, s') = \lambda R_A(s, a, s') + \gamma\Phi(s') - \Phi(s)$ for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$.

Suppose $R_A \equiv R_B$. Then there exists some $\lambda > 0$ and a bounded potential function $\Phi : \mathcal{S} \to \mathbb{R}$ such that $R_B(s, a, s') = \lambda R_A(s, a, s') + \gamma\Phi(s') - \Phi(s)$ for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. Rearranging:

$$R_A(s, a, s') = \frac{1}{\lambda} R_B(s, a, s') + \gamma \left(\frac{-1}{\lambda} \Phi(s') \right) - \left(\frac{-1}{\lambda} \Phi(s) \right). \tag{2.3}$$

Since $\frac{1}{\lambda} > 0$ and $\Phi'(s) = \frac{-1}{\lambda}\Phi(s)$ is a bounded potential function, it follows that $R_B \equiv R_A$.

Finally, suppose $R_A \equiv R_B$ and $R_B \equiv R_C$. Then there exists some $\lambda_1, \lambda_2 > 0$ and bounded potential functions $\Phi_1, \Phi_2 : \mathcal{S} \to \mathbb{R}$ such that for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$:

$$R_B(s, a, s') = \lambda_1 R_A(s, a, s') + \gamma\Phi_1(s') - \Phi_1(s), \tag{2.4}$$

$$R_C(s, a, s') = \lambda_2 R_B(s, a, s') + \gamma\Phi_2(s') - \Phi_2(s). \tag{2.5}$$

[*] Note constant shifts in the reward of an undiscounted MDP would cause the value function to diverge. Fortunately, the shaping $\gamma\Phi(s') - \Phi(s)$ is unchanged by constant shifts to Φ when $\gamma = 1$.

Substituting the expression for R_B into the expression for R_C:

$$\begin{aligned}
R_C(s,a,s') &= \lambda_2\left(\lambda_1 R_A(s,a,s') + \gamma\Phi_1(s') - \Phi_1(s)\right) + \gamma\Phi_2(s') - \Phi_2(s) && (2.6)\\
&= \lambda_1\lambda_2 R_A(s,a,s') + \gamma\left(\lambda_2\Phi_1(s') + \Phi_2(s')\right) - \left(\lambda_2\Phi_1(s) + \Phi_2(s)\right) && (2.7)\\
&= \lambda R_A(s,a,s') + \gamma\Phi(s') - \Phi(s), && (2.8)
\end{aligned}$$

where $\lambda = \lambda_1\lambda_2 > 0$ and $\Phi(s) = \lambda_2\Phi_1(s) + \Phi_2(s)$ is bounded. Thus $R_A \equiv R_C$. $\square$

If $R_A \equiv R_B$ for some fixed $(\mathcal{S}, \mathcal{A}, \gamma)$, then for any MDP\R $M^- = (\mathcal{S}, \mathcal{A}, \gamma, \mu_0, \tau)$ we have $\pi_\star\left((M^-, R_A)\right) = \pi_\star\left((M^-, R_B)\right)$, where (M^-, R) denotes the MDP specified by M^- with reward function R. In other words, R_A and R_B induce the same optimal policies for all initial state distributions μ_0 and transition dynamics τ.

Part I

Inferring agent objectives

Chapter 3

Upper bounds on reward learning

Reward learning algorithms *infer* reward functions from data. However, multiple reward functions often fit the data equally well, even in the infinite-data limit. Prior work often imposes assumptions on data sources so that reward functions can be uniquely recoverable. By contrast, we formally characterize the *partial identifiability* of reward functions inferred from popular data sources, including demonstrations and trajectory preferences, under multiple common sets of assumptions. We analyze the impact of this partial identifiability on downstream tasks such as policy optimization, including under changes in environment dynamics. We unify our results in a framework for comparing data sources and downstream tasks by their invariances, with implications for the design and selection of data sources for reward learning.

3.1 Introduction

Multiple reward functions are often consistent with a given data source, even in the infinite-data limit. For most data sources, this fundamental ambiguity has been acknowledged, but its extent has not been characterized. We contribute a formal characterization of the *ambiguity* of several popular data sources, including expert demonstrations (Section 3.3.1) and trajectory preferences (Section 3.3.2). By studying infinite-data limits, we provide an upper bound on the information recoverable from finite data sets using any algorithm. These bounds are useful for evaluating algorithms relative to their limits and choosing between different data sources.

Often, uniquely identifying a reward function is unnecessary. For example, learnt reward functions are often used for policy optimization via reinforcement learning (RL).[*] Perhaps the reward function underpinning some data is ambiguous, but all plausible reward functions already imply the same optimal policy. We suggest that the ambiguity of a data source should be evaluated *relative to* the intended downstream application. We contribute a characterization

[*] Applications also arise in other fields where reward functions are used to understand and predict the behavior of humans, animals, and other systems [see, e.g., 143, 40, 140, 66, 124, 34].

of this *ambiguity tolerance* for various applications, including policy optimization under arbitrary dynamics (Section 3.3.3).

Ambiguity and ambiguity tolerance are formally related. Both concern *invariances* — of data sources or downstream outcomes — to reward function *transformations*. This perspective provides a unifying framework in which to discuss ambiguity and ambiguity tolerance. In Section 3.4, we explore a *partial order* on reward transformation invariances and its implications for selecting and evaluating data sources, addressing an open problem in reward learning [88, §3.1].

3.1.1 Related work

Inverse reinforcement learning [IRL; 138] infers a reward function from expert demonstrations by inverting a model of the expert's *planning algorithm* [9, 146]. Existing work partially characterizes the inherent ambiguity of behavior for certain planning algorithms [112, 30] and classes of tasks [44, 80]. Using a more expressive space of reward functions that reveals novel ambiguity, we extend these results to more planning algorithms and arbitrary stochastic infinite-horizon tasks.

Another popular and effective data source is *preferences over behavioral trajectories* [3, 33]. Unlike for IRL, the ambiguity arising from these data sources has not been formally characterized. We contribute a formal characterization of the ambiguity for central models of evaluative feedback, including trajectory preferences.

Prior work has explored learning from expert behavior *and* preferences [68, 117, 22, 82], or other multimodal data sources [164, 74]. One motivation is that different data sources may provide complementary reward information [82], decreasing ambiguity. Similarly, Amin, Jiang, and Singh [6] and Cao, Cohen, and Szpruch [30] show reduced ambiguity from combining behavioral data across multiple *tasks*. Our partial order provides a general framework for understanding these results.

Computing an optimal behavioral policy is a primary application of learnt reward functions [1, 174]. Ng, Harada, and Russell [111] proved that *potential-shaping transformations* always preserve the set of optimal policies and so are always tolerable for this application. We extend this result, characterizing the full set of transformations that preserve a task's optimal policies, and considering additional policy optimization techniques such as maximum entropy RL.

Ambiguity corresponds to the *partial identifiability* [90] of the reward function modelled as a latent parameter. The prevailing response to this ambiguity in reward learning has been to impose additional constraints or assumptions until the data identifies the reward function *uniquely* (or, at least, *sufficiently* for policy optimization). Following Manski [102, 103] and Tamer [160], we instead *describe* ambiguity *given* various constraints and assumptions. This gives practitioners results appropriate for their real data (and the ambiguity tolerance of their actual application).

3.1.2 Preliminaries

We use the Markov Decision Process (MDP) formalism introduced in Section 2.1. We classify a transition (s, a, s') as *possible* in an MDP if s' is in the support of $\tau(s, a)$, otherwise it is *impossible*. A trajectory or fragment is *possible* if all of its transitions are possible and is *impossible* otherwise. A trajectory or fragment is *initial* if its first state is initial. A state or transition is *reachable* if it is part of some possible and initial trajectory.

We primarily consider return functions with various *restricted domains*, such as only *possible* or *initial* trajectories or fragments. Additionally, we focus on deterministic rewards on $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$. In Appendix A.3, we discuss how our invariance results change given alternative spaces of rewards (such as stochastic rewards, or with domain $\mathcal{S}$ or $\mathcal{S} \times \mathcal{A}$).

In addition to the optimal policies introduced in Section 2.1, we consider policies resulting from alternative objectives. Given an *inverse temperature* parameter $\beta > 0$, we define the *Boltzmann-rational* policy [131], denoted $\pi^{\star}_{\beta}$, with a softmax distribution over the optimal advantage function:

$$\pi^{\star}_{\beta}(a \mid s) = \exp\big(\beta A_{\star}(s, a)\big) / \left(\textstyle\sum_{a' \in \mathcal{A}} \exp\big(\beta A_{\star}(s, a')\big)\right) . \tag{3.1}$$

The *Maximal Causal Entropy* (MCE) policy [181, 60] is given by

$$\pi^{\mathrm{H}}_{\beta}(a \mid s) = (\exp\big(\beta Q^{\mathrm{H}}_{\beta}(s, a)\big) / \left(\textstyle\sum_{a' \in \mathcal{A}} \exp\big(\beta Q^{\mathrm{H}}_{\beta}(s, a')\big)\right) , \tag{3.2}$$

where Q^{H}_{β} is the *soft Q-function*, a regularized variant of the Q-function. Haarnoja et al. [60, Theorem 2 and Appendix A.2] show that Q^{H}_{β} is the unique function satisfying

$$Q^{\mathrm{H}}_{\beta}(s, a) = \mathbb{E}\left[R(s, a, S') + \gamma \tfrac{1}{\beta} \log \textstyle\sum_{a' \in \mathcal{A}} \exp \beta Q^{\mathrm{H}}_{\beta}(S', a')\right] . \tag{3.3}$$

The MCE policy is the result of maximizing a policy evaluation function with an entropy regularization term with weight $\alpha = \beta^{-1}$ [60]. The Boltzmann-rational policy can also be connected to a kind of (per-timestep) entropy regularization [60].

3.2 Reward function transformations

In this section, we discuss how *invariance* to reward function *transformations* relates to infinite-data ambiguity in reward learning and ambiguity tolerance in applications.

Definition 3.2.1 (Transformations and invariances)**.** A *transformation* is a map between reward functions. The *invariances* of an object X derived from reward R via function f are all transformations t that preserve f: $X = f(R) = f(t(R))$ for all R. We say that X *determines* R *up to* its invariances.

A set of transformations carves out a partition of the space of reward functions by grouping those reward functions reachable from one another using the transformations. The partition carved out by the invariances of an object is the *equivalence kernel* of the object's derivation function, grouping the reward functions from which identical objects are derived into partition blocks.

Given a reward learning data source, consider the object that encodes the information available from the data source in the infinite-data limit [90, §3.1]. The invariances of this object represent the *infinite-data ambiguity* of the data source—it is impossible to recover the reward function beyond the corresponding partition block, as that block implies indistinguishable data.

Similarly, consider a downstream application of learnt reward functions involving the computation of an object. The object's invariances capture the *ambiguity tolerance* of this computation, as by definition all reward functions in each partition block lead to identical outcomes.

We now introduce several fundamental sets of reward function transformations, forming a basis for the invariances we study in Section 3.3. We build on *potential shaping*, introduced by Ng, Harada, and Russell [111] and widely known to preserve optimal policies in all MDPs. We further distinguish special potential-shaping transformations with constant potential over an MDP's initial states.

Definition 3.2.2 (Potential Shaping)**.** A *potential function* is a function $\Phi : \mathcal{S} \to \mathbb{R}$, where $\Phi(s) = 0$ if s is a terminal state. If $\Phi(s) = k$ for all initial states, then we say that Φ is *k-initial.* Let R and R' be reward functions. Given a discount γ, we say R' is produced by *(k-initial) potential shaping* of R if $R'(s, a, s') = R(s, a, s') + \gamma \cdot \Phi(s') - \Phi(s)$ for some (k-initial) potential function Φ.

See Appendix A.1 for key properties of potential shaping. We now introduce novel transformations.

Definition 3.2.3 (S'-Redistribution)**.** Let R and R' be reward functions. Given transition dynamics τ, R' is produced by *S'-redistribution* of R if $\mathbb{E}_{S' \sim \tau(s,a)}\left[R(s, a, S')\right] = \mathbb{E}_{S' \sim \tau(s,a)}\left[R'(s, a, S')\right]$.

S'-redistribution allows changing R arbitrarily for impossible transitions. Moreover, if two states s'_1, s'_2 are in the support of $\tau(s, a)$ then S'-redistribution lets us increase $R(s, a, s'_1)$ and decrease $R(s, a, s'_2)$ by a proportionate amount. Note that S'-redistribution depends crucially on the reward function's dependence on the successor state. This set of transformations collapses to the identity for simpler spaces of reward functions, as we explore in Appendix A.3.

Definition 3.2.4 (Monotonic Transformations)**.** Let R and R' be reward functions. We say R' is produced by a *zero-preserving monotonic transformation* (ZPMT) of R if for all pairs of transitions $x, x' \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, $R(x) \leqslant R(x')$ if and only if $R'(x) \leqslant R'(x')$, and $R(x) = 0$ if and only if $R'(x) = 0$. Moreover, we say R' is produced by *positive linear scaling* of R if $R' = c \cdot R$ for a positive constant c.

The next general definition is used to describe transformations that leave a particular set of actions optimal in each state. The class is parameterized by a set-valued function $\mathcal{O}$ that specifies the optimal actions (Ψ acts as a value function). In use, we constrain $\mathcal{O}$ to $\arg\max_a A_\star(s, a)$ in some or all states s (see, e.g., Theorems 3.3.4 and 3.3.6). Thus an implicit dependence on R arises (through $A_\star$). If $\mathcal{O}$ were unconstrained, the set would contain all possible transformations.

Definition 3.2.5 (Optimality-Preserving Transformation)**.** Let R and R' be reward functions. Given a function $\mathcal{O} : \mathcal{S} \to \mathcal{P}(\mathcal{A})\backslash\{\varnothing\}$, transition dynamics τ, and discount rate γ, we say R' is produced from R by an *optimality-preserving transformation* (OPT) with $\mathcal{O}$ if there is a function $\Psi : \mathcal{S} \to \mathbb{R}$ such that $\mathbb{E}_{S' \sim \tau(s,a)}\left[R'(s, a, S') + \gamma\Psi(S')\right] \leqslant \Psi(s)$ for all s, a, with equality if and only if $a \in \mathcal{O}(s)$.

Finally, we consider transformations allowing the reward to vary freely for a given set of transitions.

Definition 3.2.6 (Masking)**.** Let R and R' be reward functions. Given a transition set $\mathcal{X} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, we say R' is produced by a *mask of* $\mathcal{X}$ from R if $R(x) = R'(x)$ for all $x \notin \mathcal{X}$.

3.3 Invariances of reward-related objects

Here, we catalogue the *invariances* of various central objects derived from reward functions (Definition 3.2.1), including expert trajectory distributions, the trajectory ranking induced by the return function, and the set of optimal policies. Some of these objects represent the information available in the infinite-data limit of a data source. Others represent the outcome of a downstream application.

If an object X can be derived from another object Y without further reference to the reward function, then X inherits Y's invariances. For example, the optimal Q-function's invariances are inherited by various expert policies. Accordingly, we organize this section by incrementally deriving objects from the reward function, cataloguing the invariances introduced in each step. This mirrors the structure of Figure 3.1. We defer proofs to Appendix A.2.

3.3.1 Invariances of expert behavior

Inverse reinforcement learning (IRL) algorithms infer a task's reward function from the behavior of task experts. Formally, this behavior is represented as an expert's *policy* or a sample of *trajectories*. To characterize policy invariances, we begin with Q-functions—which are instrumental to deriving many policies. Q-functions are invariant to S'-redistribution since they are defined as an expectation over the successor state S'. We show that this is the *only* invariance for Q-functions, and that the soft Q-function has the same invariances.

Theorem 3.3.1. *Given an MDP and a policy π, the Q-function for π, Q_π, determines R up to S'-redistribution. The optimal Q-function $Q_\star$ has the same invariances.*

Theorem 3.3.2. *Given an MDP and an inverse temperature parameter β, the soft Q-function Q^{H}_β determines R up to S'-redistribution.*

This invariance is inherited by any object that can be derived from a (soft) Q-function. However, note that S'-redistribution vanishes in simpler spaces of reward functions, as we explore in Appendix A.3.

We now turn to policies derived using various planning algorithms. These policies are instrumental in constructing the trajectories studied in IRL. For example, Ramachandran and Amir [131] assume that expert behavior is drawn from a *Boltzmann-rational* policy, and Ziebart, Bagnell, and Dey [182] assume a *Maximum Causal Entropy* policy. As these policies can be derived from $Q_\star$ and Q^{H}_β respectively, they inherit invariance to S'-redistribution. We show they are also invariant to potential shaping, but not to any other transformations. We note that the result for the MCE policy generalizes a similar result from Cao, Cohen, and Szpruch [30, Theorem 1] to MDPs with rewards dependent on successor states.

Theorem 3.3.3. *Given an MDP and an inverse temperature parameter β, the Boltzmann-rational policy $\pi^\star_\beta$, or the MCE policy π^{H}_β, determines R up to S'-redistribution and potential shaping.*

By contrast, Ng and Russell [112] and Abbeel and Ng [1] assume that experts follow an *optimal policy.* The set of optimal policies is known to be invariant to potential shaping [111] and positive linear scaling. It also inherits S'-redistribution invariance from $Q_\star$. We show that under certain assumptions, these and any additional invariances of an optimal policy are captured in a class of optimality-preserving transformations (Definition 3.2.5) based on each state's set of optimal actions.

When there are multiple optimal policies, ambiguity depends on how the given policy is selected. We do not assume full knowledge of the selection method, only the following: (1) We observe a *maximally supportive* policy, that is, one giving all optimal actions positive probability. (2) The exact probabilities in each state depend only on the set of optimal actions. We comment on how the invariance might be affected by other assumptions in Remark A.2.10.

Theorem 3.3.4. *Given an MDP, a maximally supportive optimal policy determines R up to optimality-preserving transformations with $\mathcal{O}(s) = \arg\max_a A_\star(s, a)$.*

In the infinite-data limit, trajectories sampled from a policy reveal the *distribution* of trajectories induced by the policy, and therefore *the policy itself* for all states reachable via its supported actions. Boltzmann-rational policies and MCE policies support *all* actions, so the trajectory distribution determines the policy for all *reachable* states. It follows that trajectory sampling introduces an invariance solely to changes in the reward of unreachable transitions.

Theorem 3.3.5. *Given an MDP M and an inverse temperature parameter β, the distribution of trajectories $\Delta^\star_\beta$ induced by the Boltzmann-rational policy $\pi^\star_\beta$, or $\Delta^{\mathrm{H}}_\beta$ induced by the MCE policy π^{H}_β, determines R up to S'-redistribution, potential shaping, and a mask of unreachable transitions.*

Similarly, trajectories sampled from an optimal policy reveal the policy in those states that its actions reach, and introduce invariance to transformations of reward in other states.

Theorem 3.3.6. *Given an MDP, consider the distribution of trajectories, $\Delta_\star$, induced by a maximally supportive optimal policy. Let $\mathfrak{S}$ be the set of states in supported trajectories. Let $\mathfrak{O}$ be the set of functions $\mathcal{O}$ defined on $\mathcal{S}$ such that $\mathcal{O}(s) = \arg\max_a A_\star(s,a)$ for all $s \in \mathfrak{S}$. $\Delta_\star$ determines R up to optimality-preserving transformations for any $\mathcal{O} \in \mathfrak{O}$.*

A mask of unreachable transitions is included as a special case. Note that a mask of the complement of $\mathfrak{S}$ is *not* included. However, as $\mathcal{O}$ is unconstrained outside $\mathfrak{S}$, the reward is effectively unconstrained in those states, except that the reward of transitions out of $\mathfrak{S}$ may have to "compensate" for the value of their successor states to prevent new actions that lead out of $\mathfrak{S}$ from becoming optimal.

3.3.2 Invariances of trajectory evaluation

The return function captures the reward accumulated over a trajectory and is instrumental in deriving data, such as reward labels and trajectory-preference comparisons, for evaluative feedback. Here, we consider the invariances of the return function for various restricted domains.

Theorem 3.3.7. *Given an MDP, the return function restricted to possible trajectory fragments, G_ζ, determines R up to a mask of* impossible *transitions.*

Theorem 3.3.8. *Given an MDP, the return function restricted to possible and initial trajectories, G_ξ, determines R up to zero-initial potential shaping and a mask of* unreachable *transitions.*

G_ζ has few invariances because its domain includes all possible individual transitions. Further invariances would hold given domain restrictions such as minimum or maximum fragment lengths.

Pairwise comparisons between trajectories have been studied as a data source for reward learning [3, 33]. It is common to model the comparisons as based on the return of trajectories but with accompanying *decision noise* following a *Boltzmann distribution.* Under this assumption, in the limit of infinite noisy comparisons for each pair of trajectories, the data source reveals the Boltzmann distributions governing each possible comparison. Boltzmann noise encodes the precise difference in return between trajectories, so little invariance is introduced.

Formally, given an MDP and an inverse temperature parameter $\beta > 0$, let $\preceq^{\zeta}_{\beta}$ be a stochastic relation on possible trajectory fragments, such that for each pair of possible trajectory fragments ζ_1, ζ_2,

$$\mathbb{P}(\zeta_1 \preceq^{\zeta}_{\beta} \zeta_2) = \exp(\beta G(\zeta_2)) / \left(\exp(\beta G(\zeta_1)) + \exp(\beta G(\zeta_2))\right) ,$$

and let $\preceq^{\xi}_{\beta}$ be the analogous relation on possible and initial trajectories.

Theorem 3.3.9. *Given an MDP and an inverse temperature β, the distribution of comparisons of possible trajectory fragments, $\preceq^{\zeta}_{\beta}$, determines R up to a mask of* impossible *transitions.*

Theorem 3.3.10. *Given an MDP and an inverse temperature β, the distribution of comparisons of possible and initial trajectories, $\preceq^{\xi}_{\beta}$, determines R up to k-initial potential shaping and a mask of* unreachable *transitions.*

Boltzmann comparisons of fragments have few invariances due to the broad range of comparisons permitted, including those between individual transitions and empty trajectories. Additional invariances will arise from additional restrictions, such as permitting comparisons only between fragments of a fixed length. Moreover, it is worth reiterating that these invariances rely heavily on the precise structure of the decision noise revealing cardinal information in the infinite-data limit.

We also consider noiseless comparisons of the return. The infinite-data limit then corresponds to the order induced by the return function. Formally, define the *noiseless order of possible trajectory fragments* as a relation, $\preceq^{\zeta}_{\star}$, on possible trajectory fragments: $\zeta_1 \preceq^{\zeta}_{\star} \zeta_2 \Leftrightarrow G(\zeta_1) \leqslant G(\zeta_2)$. Similarly, define the *noiseless order of possible and initial trajectories* as the analogous relation, $\preceq^{\xi}_{\star}$, for pairs of *possible* and *initial* trajectories. These relations omit cardinal information about pairwise comparisons, so invariances to certain monotonic transformations are introduced. The precise monotonic invariances depend on the MDP (for example, see the proof in Appendix A.2.4).

Theorem 3.3.11. *We have the following bounds on the invariances of the noiseless order of possible trajectory fragments, $\preceq^{\zeta}_{\star}$. In all MDPs:*

(1) $\preceq^{\zeta}_{\star}$ is invariant to positive linear scaling and a mask of impossible transitions; and

(2) $\preceq^{\zeta}_{\star}$ is not invariant to transformations other than zero-preserving monotonic transformations or masks of impossible transitions.

Moreover, there exist MDPs attaining each of these bounds.

We give a lower bound on the invariances of the noiseless order of possible and initial trajectories, $\preceq^{\xi}_{\star}$, inherited from $\preceq^{\xi}_{\beta}$ and $\preceq^{\zeta}_{\star}$. Note that this lower bound does *not* rule out additional invariances (unlike our other results). We comment further in Appendix A.2.4.

Theorem 3.3.12. *Given an MDP, the noiseless order of possible and initial trajectories, $\preceq^{\xi}_{\star}$, is invariant to (at least) k-initial potential shaping, positive linear scaling, and a mask of unreachable transitions.*

We next give the transformations that preserve preferences over *lotteries* of trajectories. It is possible to model such preferences as von Neumann–Morgenstern (VNM)-rational choices between lotteries (distributions) over trajectory returns. Such lotteries are well known to be invariant to positive affine transformations of return [110, appendix A]. We show that these transformations correspond to k-initial potential shaping and positive linear scaling of the reward function. Let $\preceq^{\xi}_{\mathcal{D}}$ be the relation on distributions over possible initial trajectories: $\mathcal{D}_1 \preceq^{\xi}_{\mathcal{D}} \mathcal{D}_2 \Leftrightarrow \mathbb{E}_{\Xi\sim\mathcal{D}_1}\left[G(\Xi)\right] \leqslant \mathbb{E}_{\Xi\sim\mathcal{D}_2}\left[G(\Xi)\right]$.

Theorem 3.3.13. *Given an MDP, $\preceq^{\xi}_{\mathcal{D}}$ determines R up to k-initial potential shaping, positive linear scaling, and a mask of unreachable transitions.*

3.3.3 Invariances of policy optimization

The primary application of learnt reward functions is to compute optimal policies using techniques such as RL. Policy optimization procedures typically compute a single optimal policy. However, in terms of invariances, one may desire to preserve the whole *set* of optimal policies so as not to tolerate any suboptimal policies *becoming* optimal through a reward transformation.

The set of optimal policies inherits S'-redistribution invariance from the optimal Q-function and is also known to be invariant to potential shaping [111]. In fact, because a maximally supportive optimal policy can be derived from the set of optimal policies *and vice versa*, the set shares exactly the same invariances as a maximally supportive optimal policy (Theorem 3.3.4).

Theorem 3.3.14. *Given an MDP, the set of optimal policies determines R up to optimality-preserving transformations with $\mathcal{O}(s) = \arg\max_a A_\star(s, a)$.*

Moreover, if one uses an algorithm not guaranteed to find a globally optimal policy, one may desire to preserve the entire *order* induced on the space of policies by the policy evaluation function, rather than just the set of maximizing policies. Future work could investigate the invariances of the ordinal information in the policy evaluation function. Note that since the set of optimal policies can be derived from this order, the order has at most the invariances of the set of optimal policies.

3.4 Implications for reward learning

So far, we have catalogued the *invariances* of various objects to transformations of the reward function. These invariances characterize the *infinite-data ambiguity* of several reward learning data sources and the *ambiguity tolerance* of policy optimization. In this section, we discuss the implications for practical evaluation of reward learning data sources.

We begin by defining a framework for comparing data sources and applications based on their ambiguity. The characterization of ambiguity and tolerance as invariances to reward

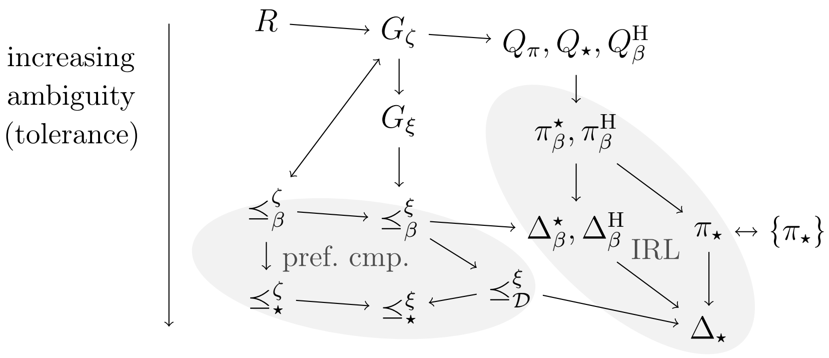

increasing
ambiguity
(tolerance)
R
G_ζ
$Q_\pi, Q_\star, Q^{\mathrm{H}}_\beta$
G_ξ
$\pi^\star_\beta, \pi^{\mathrm{H}}_\beta$
$\preceq^\zeta_\beta$
$\preceq^\xi_\beta$
$\Delta^\star_\beta, \Delta^{\mathrm{H}}_\beta$
$\pi_\star \leftrightarrow \{\pi_\star\}$
pref. cmp.
IRL
$\preceq^\xi_{\mathcal{D}}$
$\preceq^\zeta_\star$
$\preceq^\xi_\star$
$\Delta_\star$

Figure 3.1: The *infinite-data ambiguity* of reward learning data sources and the *ambiguity tolerance* of downstream applications of a learnt reward function are both *invariances* of objects derived from reward functions (Definition 3.2.1). These invariances imbue the data sources and applications with a partial order by *ambiguity refinement* (Definition 3.4.1). For a fixed MDP, here we give the partial order. $X \to Y$ means $X \preceq Y$, that is, X is no more ambiguous than Y as a data source (or, as a downstream application, Y is tolerant to X's ambiguity). Note that the partial order is transitive—X is no more ambiguous than Y (Y is tolerant to X's ambiguity) if there is a path from X to Y. These objects are defined in Sections 3.1.2 and 3.3.

transformations suggests a natural *partial order* on data sources and applications. Recall that the invariances of an object correspond to a *partition* of the space of reward functions (Section 3.2). We lift the *refinement relation* for partitions [2, §I.2.B] to data sources and applications as follows.

Definition 3.4.1 (Ambiguity refinement)**.** Consider two reward learning data sources (or applications) X and Y, and let Π_X and Π_Y be the partitions of the space of reward functions corresponding to their invariances (Definition 3.2.1). If Π_X is a partition refinement of Π_Y, we write $X \preceq Y$ and say X is *no more ambiguous* than Y (or X is *tolerable for* application Y). If $X \preceq Y$ but not $Y \preceq X$, then we write $X \prec Y$ and say X is (strictly) *less ambiguous* than Y.

Given two data sources X and Y, $X \preceq Y$ corresponds to X conflating no additional reward functions compared to Y in the infinite-data limit. This is the sense in which we say X is *no more ambiguous* than Y. Moreover, given a downstream application Z, $X \preceq Z$ is precisely the condition of Z tolerating the infinite-data ambiguity of data source X: $X \preceq Z$ if and only if the reward functions conflated by X in the infinite-data limit all lead to the same outcome in Z.

More concretely, we can compare the ambiguity of specific data sources and applications. Figure 3.1 depicts the partial order for a fixed MDP. For example, the ambiguity tolerance of the set of optimal policies is a class of optimality-preserving transformations. Data sources

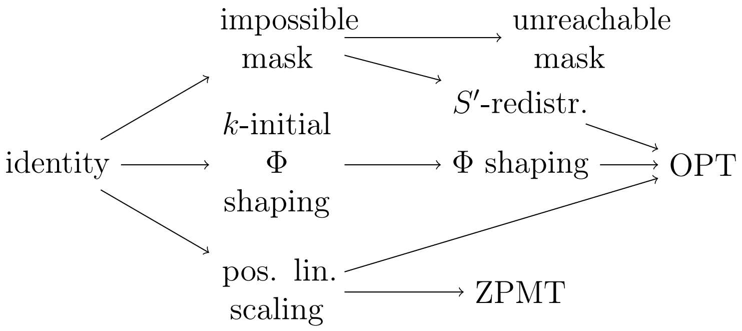


Figure 3.2: For a fixed MDP, here we give the partition refinement relation for each of the basic families of reward transformations defined in Section 3.2. Each of our ambiguity results involve compositions of these basic invariances. This order is therefore helpful for calculating ambiguity refinements.

that are less ambiguous than this tolerance (higher in Figure 3.1) are sufficient for policy optimization. Notably, this excludes noiseless comparisons between trajectory fragments in some MDPs. Specifically, policy optimization does not, in general, tolerate zero-preserving monotonic transformations (ZPMTs), while noiseless comparisons are invariant to ZPMTs in some MDPs (Theorem 3.3.11). Policy optimization is also intolerant to data sources based on possible and initial trajectories, which are invariant to a mask of unreachable transitions. However, these sources are tolerable if the application only requires optimal behavior in reachable states.

Moreover, we can compare data sources drawn from one MDP to applications in *another* MDP, such as under a shift in transition dynamics or initial state distribution. This captures the common sim-to-real setting where learning occurs in a simulated or otherwise restricted environment that differs from the deployment environment. The simplest transformations to consider are masks of impossible or unreachable transitions — these depend on transition dynamics. In general, the ambiguity corresponding to a mask of $\mathcal{X}$ is less than for a mask of $\mathcal{X}' \supset \mathcal{X}$. For example, if the new dynamics supports previously impossible transitions, then sources with invariance to an impossible-transition mask from the original MDP may not be tolerable for applications in the new MDP.

A similar result holds for S'-redistribution, which involves an expectation over MDP dynamics. Moreover, the effect of S'-redistribution on optimal behavior under changed dynamics can be dramatic. For example, if τ changes for each state-action pair, then $Q_\star$ is completely undetermined, which means that S'-redistribution could make *any policy whatsoever* seem optimal under the new transition dynamics. This is a consequence of the following general result: if we determine R modulo S'-redistribution under τ, and then shift to some other transition dynamics τ', then $\mathbb{E}_{S' \sim \tau'(s,a)}\left[R'(s,a,S')\right]$ is completely undetermined for all s, a where $\tau(s,a) \neq \tau'(s,a)$. Note that this result relies on the formulation of rewards as depending on the successor state (see Appendix A.3).

Theorem 3.4.2. *Consider an MDP $(\mathcal{S}, \mathcal{A}, \tau, \mu_0, R_1, \gamma)$ and alternative transition dynamics τ'. Given any function $\mathcal{L} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, there exists a reward function R_2, produced from R_1 by S'-redistribution under τ, such that $\mathbb{E}_{S' \sim \tau'(s,a)}\left[R_2(s, a, S')\right] = \mathcal{L}(s, a)$ for all s, a such that $\tau(s, a) \neq \tau'(s, a)$.*

Ambiguity refinement is a *partial* order, and some data sources are indeed *incomparable*. In consolation, we observe that such incomparable ambiguity is *complementary* ambiguity, in that by combining the associated data sources, we reduce overall ambiguity about the latent reward.

Theorem 3.4.3. *Given data sources X and Y, let (X, Y) denote the combined data source formed from X and Y. If X and Y are incomparable, then $(X, Y) \prec X$ and $(X, Y) \prec Y$.*

This perspective highlights promising directions for the design of reward learning data sources. In particular, it suggests developing reward learning algorithms for mixtures of data sources with *complementary* ambiguity. Most popular data sources appear to have similar kinds of ambiguity given a fixed MDP. However, ambiguity could be reduced by incorporating data from *multiple MDPs* with different dynamics or discount rates. This is a new perspective in which to frame some of the results of Amin, Jiang, and Singh [6] and Cao, Cohen, and Szpruch [30].

3.5 Limitations and future work

Our results give an upper bound on the amount of information that can be extracted from various data sources by studying the limit of infinite data. In practice, these limits may never be attained on finite data sets. Future work should characterize how much information is contained in data sets of varying sizes. This would determine the most efficient data source for a fixed data-collection budget.

It is important to choose the right reward learning algorithm when designing an automated system. Learning the wrong reward function can cause large negative impacts [8], especially for systems of increased power [23, §12]. We stress that system designers must evaluate reward learning approaches holistically, with our work on ambiguity contributing to one dimension. In the following, we describe two complementary axes for evaluating reward learning algorithms.

Our results apply under the assumptions made by popular reward learning algorithms (see Section 3.3). However, these assumptions may not be *sound* for real data. In fact, there are important differences between human data and data synthesized with standard assumptions [115]. Moreover, there can be a trade-off between ambiguity and plausibility: a data source may have low ambiguity *because* it makes unrealistic assumptions. At one extreme, directly requesting a human's reward function leaves no ambiguity, but is unsound, this being the reason we set out to learn rewards.

Furthermore, even given an ideal reward learning algorithm, *what* kinds of rewards should we seek to learn—stated preferences, revealed preferences, instructions, or something else [50]? *Who* should we seek to learn rewards from? Such *normative* questions may constrain our choice of data sources. All stakeholders should be considered in the design of automated systems. Perhaps only certain experts can provide demonstrations, but more stakeholders can provide comparisons.

3.6 Conclusion

Substantial effort has been invested to develop reward learning algorithms for a variety of data sources. A fundamental question is how effective these algorithms are, relative to an optimal algorithm for that data source? Our contribution is to characterize the information available from each data source, thereby establishing an upper bound on the performance of any algorithm using that data source.

In particular, we prove invariances of various reward-related objects to transformations such as potential shaping. Moreover, we show that these objects form a partial order under *ambiguity refinement*. The resulting framework enables direct comparisons between data sources. We find that some data sources contain strictly less information than others, such as noiseless preference comparisons vs. return labels. By contrast, others are incomparable and have complementary ambiguity, such as Q-values and trajectory returns.

While practitioners could simply collect data from the least ambiguous source, this might be costly. We also characterize the ambiguity tolerance of downstream applications (such as policy optimization) for which the reward function is used. This enables practitioners to identify data sources that are precise only in the areas their application needs, limiting unnecessary costs.

Chapter 4

Distance metrics on reward functions

Prior work has evaluated learned reward functions by evaluating policies optimized for the learned reward. However, this method cannot distinguish between the learned reward function failing to reflect user preferences and the policy optimization process failing to optimize the learned reward. Moreover, this method can only tell us about behavior in the evaluation environment, but the reward may incentivize very different behavior in even a slightly different deployment environment. To address these problems, we introduce the *Equivalent-Policy Invariant Comparison (EPIC)* distance to quantify the difference between two reward functions directly, without a policy optimization step. We prove EPIC is invariant on an equivalence class of reward functions that always induce the same optimal policy. Furthermore, we find EPIC can be efficiently approximated and is more robust than baselines to the choice of coverage distribution. Our source code is available at `https://github.com/HumanCompatibleAI/evaluating-rewards`.

4.1 Introduction

Prior work has usually evaluated the learned reward function $\hat{R}$ using the "rollout method": training a policy $\pi_{\hat{R}}$ to optimize $\hat{R}$ and then examining rollouts from $\pi_{\hat{R}}$. Unfortunately, using RL to compute $\pi_{\hat{R}}$ is often computationally expensive. Furthermore, the method produces *false negatives* when the reward $\hat{R}$ matches user preferences but the RL algorithm fails to optimize with respect to $\hat{R}$.

The rollout method also produces *false positives*. Of the many reward functions that induce the desired rollout in a given environment, only a small subset align with the user's preferences. For example, suppose the agent can reach states $\{A, B, C\}$. If the user prefers $A > B > C$, but the agent instead learns $A > C > B$, the agent will still go to the correct state A. However, if the initial state distribution or transition dynamics change, misaligned rewards may induce undesirable policies. For example, if A is no longer reachable at deployment, the previously reliable agent would misbehave by going to the least-favored state C.

Table 4.1: Summary of the desiderata satisfied by each reward function distance studied in this dissertation. **Key** — the distance is: a *pseudometric* (Definition 4.2.1); *invariant* to potential shaping [111] and positive rescaling (Section 2.2); a computationally *efficient* approximation achieving low error (Section 4.4.1); *robust* to the choice of coverage distribution (Section 4.4.2); and *predictive* of the similarity of the trained policies (Section 5.2).

Distance	Pseudometric	Invariant	Efficient	Robust	Predictive
EPIC	✓	✓	✓	✓	✓
NPEC	✗	✓	✗	✗	✓
ERC	✓	✗	✓	✗	✓

We propose instead to evaluate learned rewards via their distance from other reward functions. Table 4.1 summarizes our desiderata for reward function distances.

For benchmarks, it is usually possible to directly compare a learned reward $\hat{R}$ to the true reward function R. Alternatively, benchmark creators can train a "proxy" reward function from a large human data set. This proxy can then be used as a stand-in for the true reward R when evaluating algorithms trained on a different or smaller data set.

Comparison with a ground-truth reward function is rarely possible outside of benchmarks. However, even in this challenging case, comparisons can at least be used to cluster reward models trained using different techniques or data. Larger clusters are more likely to be correct, since multiple methods arrived at a similar result. Moreover, our regret bound (Theorem 5.1.1) suggests we could use interpretability methods discussed in Chapter 6 on one model and get some guarantees for models in the same cluster.

To the best of our knowledge, there is no prior work that focuses on evaluating reward functions directly. The most closely related work is Ng, Harada, and Russell [111], identifying reward transformations guaranteed to not change the optimal policy. However, a variety of ad hoc methods have been developed to evaluate reward functions. The rollout method—evaluating rollouts of a policy trained on the learned reward—is evident in the earliest work on IRL [112]. Fu, Luo, and Levine [49] refined the rollout method by testing on a transfer environment, inspiring our experiment in Section 5.2. Recent work has compared reward functions by scatterplotting returns [68, 27], inspiring our Episode Return Correlation (ERC) baseline (Section 4.3.1).

We introduce the *Equivalent-Policy Invariant Comparison (EPIC)* distance that meets all the criteria in Table 4.1. We believe EPIC is the first method to quantitatively evaluate reward functions without training a policy. EPIC (Section 4.2) canonicalizes the reward functions' potential-based shaping [111], then takes the correlation between the canonical rewards over a *coverage distribution* $\mathcal{D}$ of transitions. We also introduce baselines *Nearest Point in Equivalence Class (NPEC)* and *ERC* (Section 4.3) which partially satisfy the criteria.

EPIC works best when $\mathcal{D}$ has support on all realistic transitions. We achieve this in our experiments by using uninformative priors, such as rollouts of a policy taking random actions.

Moreover, we find that EPIC is robust to the exact choice of distribution $\mathcal{D}$, producing similar results across a range of distributions, whereas ERC and especially NPEC are highly sensitive to the choice of $\mathcal{D}$ (Section 4.4.2).

Moreover, low EPIC distance between a learned reward $\hat{R}$ and the true reward R predicts low regret. That is, the policies $\pi_{\hat{R}}$ and π_R optimized for $\hat{R}$ and R obtain similar returns under R. Theorem 5.1.1 can be used to bound the regret even in unseen environments; by contrast, the rollout method can only determine regret in the evaluation environment. We also confirm this result empirically (Section 5.2).

4.2 Comparing reward functions with EPIC

In this section, we introduce the *Equivalent-Policy Invariant Comparison (EPIC)* pseudometric. This novel distance canonicalizes the reward functions' potential-based shaping, then compares the canonical representatives using Pearson correlation, which is invariant to scale. Together, this construction makes EPIC invariant on reward equivalence classes. See Section B.3.1 for proofs.

First, we must define the notion of a distance.

Definition 4.2.1. Let X be a set and $d : X \times X \to [0, \infty)$ a function. d is a *premetric* if $d(x, x) = 0$ for all $x \in X$. d is a *pseudometric* if, furthermore, it is symmetric, $d(x, y) = d(y, x)$ for all $x, y \in X$, and satisfies the triangle inequality, $d(x, z) \leqslant d(x, y) + d(y, z)$ for all $x, y, z \in X$. d is a *metric* if, furthermore, for all $x, y \in X$, $d(x, y) = 0 \implies x = y$.

We wish for $d(R_A, R_B) = 0$ whenever the rewards are equivalent (Definition 2.2.2), $R_A \equiv R_B$, even if they are not identical, $R_A \neq R_B$. This is forbidden in a metric but permitted in a pseudometric, while retaining other guarantees such as symmetry and triangle inequality that a metric provides. Accordingly, a pseudometric is usually the best choice for a distance d over reward functions.

We define the *canonically shaped reward* $C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R)$ as an expectation over some arbitrary distributions $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{A}$ over states $\mathcal{S}$ and actions $\mathcal{A}$ respectively. These two distributions can be combined to define a distribution over transitions by taking their outer product, $\mathcal{D}_\mathcal{S} \times \mathcal{D}_\mathcal{A} \times \mathcal{D}_\mathcal{S}$, which we refer to as the *canonicalization distribution.* This construction means that $C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R)$ does not depend on the MDP's initial state distribution μ_0 or transition dynamics τ. In particular, we may evaluate R on transitions that are impossible in the training environment, since these may become possible in a deployment environment with a different μ_0 or τ.

Definition 4.2.2 (Canonically Shaped Reward)**.** Let $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be a reward function. Given distributions $\mathcal{D}_\mathcal{S} \in \Delta(\mathcal{S})$ and $\mathcal{D}_\mathcal{A} \in \Delta(\mathcal{A})$ over states and actions, let S and S' be random variables independently sampled from $\mathcal{D}_\mathcal{S}$ and let A be sampled from $\mathcal{D}_\mathcal{A}$. We define the *canonically shaped* R to be

$$\begin{aligned} C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R\right)\left(s,a,s'\right) = {} & R(s,a,s') \\ & + \mathbb{E}\left[\gamma R(s',A,S') - R(s,A,S') - \gamma R(S,A,S')\right]. \end{aligned} \tag{4.1}$$

Informally, if R is shaped by potential Φ, then increasing $\Phi(s)$ decreases $R(s,a,s')$ but increases $\mathbb{E}\left[-R(s,A,S')\right]$, canceling. Similarly, increasing $\Phi(s')$ increases $R(s,a,s')$ but decreases $\mathbb{E}\left[\gamma R(s',A,S')\right]$. Finally, $\mathbb{E}[\gamma R(S,A,S')]$ centers the reward, canceling constant shift.

Proposition 4.2.3 (The Canonically Shaped Reward is Invariant to Shaping)**.** *Let $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be a reward function and $\Phi : \mathcal{S} \to \mathbb{R}$ a potential function. Let $\gamma \in [0,1]$ be a discount rate, and $\mathcal{D}_{\mathcal{S}} \in \Delta(\mathcal{S})$ and $\mathcal{D}_{\mathcal{A}} \in \Delta(\mathcal{A})$ be distributions over states and actions. Let R' denote R shaped by Φ: $R'(s,a,s') = R(s,a,s') + \gamma\Phi(s') - \Phi(s)$. Then the canonically shaped R' and R are equal: $C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R'\right) = C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R\right)$.*

Proof. See Section B.3.1. □

Proposition 4.2.3 holds for arbitrary distributions $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{A}}$. However, in the following proposition we show that the potential shaping applied by the canonicalization $C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R\right)$ is more influenced by perturbations to R of transitions (s,a,s') with high joint probability. This suggests choosing $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{A}}$ to have broad support, making $C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R\right)$ more robust to perturbations of any given transition.

Proposition 4.2.4. *Let $\mathcal{S}$ and $\mathcal{A}$ be finite, with $|\mathcal{S}| \geqslant 2$. Let $\mathcal{D}_{\mathcal{S}} \in \Delta(\mathcal{S})$ and $\mathcal{D}_{\mathcal{A}} \in \Delta(\mathcal{A})$. Let R, $\nu : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be reward functions, with $\nu(s,a,s') = \lambda\mathbb{I}[(s,a,s') = (x,u,x')]$, $\lambda \in \mathbb{R}$, $x, x' \in \mathcal{S}$, and $u \in \mathcal{A}$. Let $\Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)(s,a,s') = C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R\right)\left(s,a,s'\right) - R(s,a,s')$. Then,*

$$\left\|\Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R+\nu) - \Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)\right\|_{\infty} = \lambda\left(1 + \gamma\mathcal{D}_{\mathcal{S}}(x)\right)\mathcal{D}_{\mathcal{A}}(u)\mathcal{D}_{\mathcal{S}}(x'). \tag{4.2}$$

We have canonicalized potential shaping; next, we compare the rewards in a scale-invariant manner.

Definition 4.2.5. The *Pearson distance* between random variables X and Y is defined by the expression $D_\rho(X,Y) = \sqrt{1-\rho(X,Y)}/\sqrt{2}$, where $\rho(X,Y)$ is the Pearson correlation between X and Y.

Lemma 4.2.6. *The Pearson distance D_ρ is a pseudometric. Moreover, let $a, b \in (0,\infty)$, $c, d \in \mathbb{R}$, and X, Y be random variables. Then it follows that $0 \leqslant D_\rho(aX+c, bY+d) = D_\rho(X,Y) \leqslant 1$.*

We can now define EPIC in terms of the Pearson distance between canonically shaped rewards.

Definition 4.2.7 (Equivalent-Policy Invariant Comparison (EPIC) pseudometric)**.** Let $\mathcal{D}$ be some coverage distribution over transitions $s \xrightarrow{a} s'$. Let S, A, S' be random variables jointly sampled from $\mathcal{D}$. Let $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{A}}$ be some distributions over states $\mathcal{S}$ and $\mathcal{A}$ respectively. The *Equivalent-Policy Invariant Comparison (EPIC)* distance between reward functions R_A and R_B is

$$D_{\text{EPIC}}(R_A, R_B) = D_\rho\left(C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R_A\right)(S, A, S'), C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R_B\right)(S, A, S')\right). \tag{4.3}$$

Theorem 4.2.8. *The Equivalent-Policy Invariant Comparison distance is a pseudometric.*

Since EPIC is a pseudometric, it satisfies the triangle inequality. To see why this is useful, consider an environment with an expensive-to-evaluate ground-truth reward R. Directly comparing many learned rewards $\hat{R}$ to R might be prohibitively expensive. We can instead pay a one-off cost: query R a finite number of times and infer a proxy reward R_P with $D_{\text{EPIC}}(R, R_P) \leqslant \epsilon$. The triangle inequality allows us to evaluate $\hat{R}$ via comparison to R_P, since $D_{\text{EPIC}}(\hat{R}, R) \leqslant D_{\text{EPIC}}(\hat{R}, R_P) + \epsilon$. This is particularly useful for benchmarks, which can be expensive to build but should be cheap to use.

Theorem 4.2.9. *Let R_A, R'_A, $R_B, R'_B : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be reward functions such that $R'_A \equiv R_A$ and $R'_B \equiv R_B$. Then $0 \leqslant D_{\text{EPIC}}(R'_A, R'_B) = D_{\text{EPIC}}(R_A, R_B) \leqslant 1$.*

The following is our main theoretical result, showing that the $D_{\text{EPIC}}(R_A, R_B)$ distance gives an upper bound on the difference in returns under *either* R_A or R_B between optimal policies $\pi_\star^{R_A}$ and $\pi_\star^{R_B}$. In other words, EPIC bounds the regret under R_A of using $\pi_\star^{R_B}$ instead of $\pi_\star^{R_A}$. Moreover, by symmetry $D_{\text{EPIC}}(R_A, R_B)$ also bounds the regret under R_B of using $\pi_\star^{R_A}$ instead of $\pi_\star^{R_B}$.

To demonstrate EPIC's properties, we compare the gridworld reward functions from Figure 4.1, reporting the distances between all reward pairs in Figure B.2. `Dense` is a rescaled and shaped version of `Sparse`, despite looking dissimilar at first glance, so $D_{\text{EPIC}}(\texttt{Sparse}, \texttt{Dense}) = 0$. By contrast, $D_{\text{EPIC}}(\texttt{Path}, \texttt{Cliff}) = 0.27$. In *deterministic* gridworlds, `Path` and `Cliff` have the same optimal policy, so the rollout method could wrongly conclude they are equivalent. But in fact the rewards are fundamentally different: when there is a significant risk of "slipping" in the wrong direction, the optimal policy for `Cliff` walks along the top instead of the middle row, incurring a -1 penalty to avoid the risk of falling into the -4 "cliff."

For this example, we used state and action distributions $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{A}}$ uniform over $\mathcal{S}$ and $\mathcal{A}$, and coverage distribution $\mathcal{D}$ uniform over state-action pairs (s, a), with s' deterministically computed. It is important these distributions have adequate support. As an extreme example, if $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}$ have no support for a particular state, then the reward of that state has no effect on the distance. We can compute EPIC exactly in a tabular setting, but in general, we use a sample-based approximation (Section B.1.1).

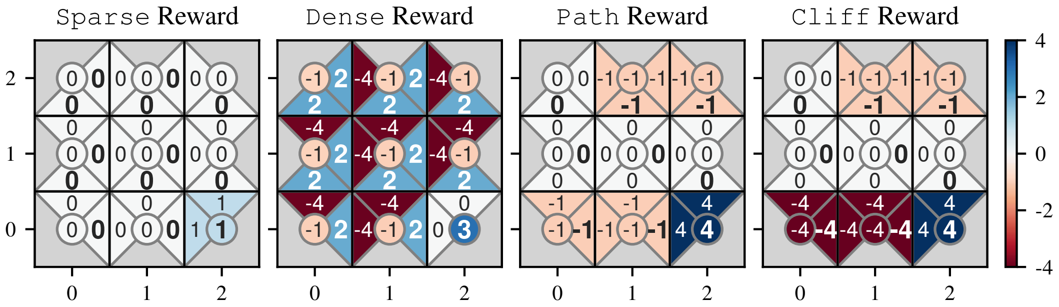


Figure 4.1: Heatmaps of four reward functions for a 3 × 3 gridworld. `Sparse` and `Dense` look different but are actually equivalent with $D_{\text{EPIC}}(\texttt{Sparse}, \texttt{Dense}) = 0$. By contrast, the optimal policies for `Path` and `Cliff` are the same if the gridworld is deterministic but different if it is "slippery." EPIC recognizes this difference with $D_{\text{EPIC}}(\texttt{Path}, \texttt{Cliff}) = 0.27$. **Key**: Reward $R(s, s')$ for moving from s to s' is given by the **triangular wedge** in cell s that is adjacent to cell s'. $R(s, s)$ is given by the **central circle** in cell s. Optimal action(s) (deterministic, infinite horizon, discount $\gamma = 0.99$) have **bold** labels. See Figure B.2 for the distances between all reward pairs.

4.3 Baseline approaches for comparing reward functions

Given the lack of established methods, we develop two alternatives as baselines: Episode Return Correlation (ERC) and Nearest Point in Equivalence Class (NPEC).

4.3.1 Episode Return Correlation (ERC)

The goal of an MDP is to maximize expected episode return, so it is natural to compare reward functions by the returns they induce. If the return of a reward function R_A is a positive affine transformation of another reward R_B, then R_A and R_B have the same set of optimal policies. This suggests using Pearson distance, which is invariant to positive affine transformations.

Definition 4.3.1 (Episode Return Correlation (ERC) pseudometric)**.** Let $\mathcal{D}$ be some distribution over trajectories. Let E be a random variable sampled from $\mathcal{D}$. The *Episode Return Correlation* distance between reward functions R_A and R_B is the Pearson distance between their episode returns on $\mathcal{D}$, $D_{\text{ERC}}(R_A, R_B) = D_\rho(g(E; R_A), g(E; R_B))$.

Prior work has produced scatter plots of the return of R_A against R_B over episodes [27, Figure 3] and fixed-length segments [68, section D]. ERC is the Pearson distance of such plots, so it is a natural baseline. We approximate ERC by the correlation of episode returns on a finite collection of rollouts.

ERC is invariant to shaping when the initial state s_0 and terminal state s_T are fixed. Let R be a reward function and Φ a potential function, and define the shaped reward

$R'(s, a, s') = R(s, a, s') + \gamma\Phi(s') - \Phi(s)$. The return under the shaped reward on a trajectory $\tau = (s_0, a_0, \cdots, s_T)$ is $g(\tau; R') = g(\tau; R) + \gamma^T\Phi(s_T) - \Phi(s_0)$. Since s_0 and s_T are fixed, $\gamma^T\Phi(s_T) - \Phi(s_0)$ is constant. It follows that ERC is invariant to shaping, as Pearson distance is invariant to constant shifts. In fact, for infinite-horizon discounted MDPs, only s_0 needs to be fixed, since $\gamma^T\Phi(s_T) \to 0$ as $T \to \infty$.

However, if the initial state s_0 is stochastic, then the ERC distance can take on arbitrary values under shaping. Let R_A and R_B be two arbitrary reward functions. Suppose that there are at least two distinct initial states, s_X and s_Y, with non-zero measure in $\mathcal{D}$. Choose potential $\Phi(s) = 0$ everywhere except $\Phi(s_X) = \Phi(s_Y) = c$, and let R'_A and R'_B denote R_A and R_B shaped by Φ. As $c \to \infty$, the correlation $\rho\left(g(E; R'_A), g(E; R'_B)\right) \to 1$. This is because the relative difference tends to zero, even though $g(E; R'_A)$ and $g(E; R'_B)$ continue to have the same absolute difference as c varies. Consequently, the ERC pseudometric $D_{\text{ERC}}(R'_A, R'_B) \to 0$ as $c \to \infty$. By an analogous argument, setting $\Phi(s_X) = c$ and $\Phi(s_Y) = -c$ gives $D_{\text{ERC}}(R'_A, R'_B) \to 1$ as $c \to \infty$.

4.3.2 Nearest Point in Equivalence Class (NPEC)

NPEC takes the minimum L^p distance between equivalence classes. See Section B.3.2 for proofs.

Definition 4.3.2 (L^p distance)**.** Let $\mathcal{D}$ be a coverage distribution over transitions $s \xrightarrow{a} s'$ and let $p \geqslant 1$ be a power. The L^p *distance* between reward functions R_A and R_B is the L^p norm of their difference:

$$D_{L^p,\mathcal{D}}(R_A, R_B) = \left(\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[|R_A(s, a, s') - R_B(s, a, s')|^p\right]\right)^{1/p}. \tag{4.4}$$

The L^p distance is affected by potential shaping and positive rescaling that do not change the optimal policy. A natural solution is to take the distance from the *nearest point* in the equivalence class: $D^U_{\text{NPEC}}(R_A, R_B) = \inf_{R'_A \equiv R_A} D_{L^p,\mathcal{D}}(R'_A, R_B)$. Unfortunately, D^U_{NPEC} is sensitive to R_B's scale.

It is tempting to instead take the infimum over both arguments of $D_{L^p,\mathcal{D}}$. However, $\inf_{R'_A \equiv R_A, R'_B \equiv R_B} D_{L^p,\mathcal{D}}(R'_A, R_B) = 0$ since all equivalence classes come arbitrarily close to the origin in L^p space. Instead, we fix this by normalizing D^U_{NPEC}.

Definition 4.3.3. *NPEC* is defined by

$$D_{\text{NPEC}}(R_A, R_B) = D^U_{\text{NPEC}}(R_A, R_B)/D^U_{\text{NPEC}}(\texttt{Zero}, R_B) \tag{4.5}$$

when $D^U_{\text{NPEC}}(\texttt{Zero}, R_B) \neq 0$, and is otherwise given by $D_{\text{NPEC}}(R_A, R_B) = 0$.

If $D^U_{\text{NPEC}}(\texttt{Zero}, R_B) = 0$, then $D^U_{\text{NPEC}}(R_A, R_B) = 0$ since R_A can be scaled arbitrarily close to $\texttt{Zero}$. Since all policies are optimal for $R \equiv \texttt{Zero}$, we choose $D_{\text{NPEC}}(R_A, R_B) = 0$ in this case.

Theorem 4.3.4. *D_{NPEC} is a premetric on the space of bounded reward functions. Moreover, let $R_A, R_A{}', R_B, R_B{}' : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be bounded reward functions such that $R_A \equiv R_A{}'$ and $R_B \equiv R_B{}'$. Then $0 \leqslant D_{\text{NPEC}}(R_A{}', R_B{}') = D_{\text{NPEC}}(R_A, R_B) \leqslant 1$.*

Proof. Pseudometric follows from $D_{L^p,\mathcal{D}}$ a pseudometric; see Section B.3.2 for details. The invariance to $R'_A \equiv R_A$ is immediate from the infimum being over $R \equiv R_A$. The invariance to $R'_B \equiv R_B$ is due to translational invariance of $D_{L^p,\mathcal{D}}$, and

$$D^U_{\text{NPEC}}(R_A, \lambda R_B) = \lambda D^U_{\text{NPEC}}(R_A, R_B), \text{ for } \lambda > 0.$$

Upper bound of 1 is due to

$$D^U_{\text{NPEC}}(R_A, R_B) \leqslant D^U_{\text{NPEC}}(\texttt{Zero}, R_B),$$

while lower bound is immediate from $D_{L^p,\mathcal{D}}$ being non-negative. See section B.3.2 for details. □

Note that D_{NPEC} may not be symmetric and so is not, in general, a pseudometric: see proposition B.3.3. The infimum in D^U_{NPEC} can be computed exactly in a tabular setting, but in general we must approximate it using gradient descent. This gives an upper bound for D^U_{NPEC}, but the quotient of upper bounds D_{NPEC} may be too low or too high. See Section B.1.2 for details of the approximation.

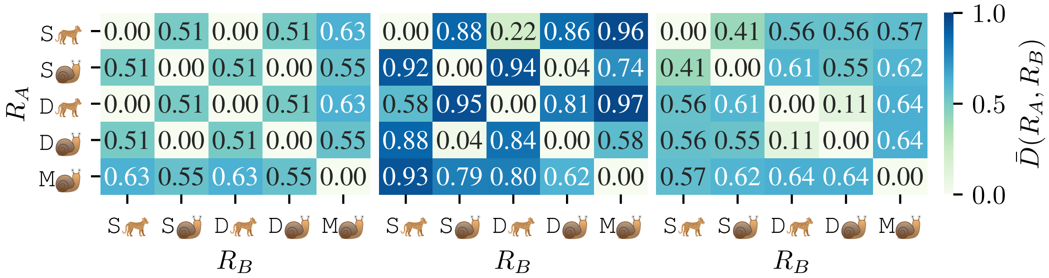
4.4 Experiments

We evaluate EPIC and the baselines ERC and NPEC in a variety of continuous control tasks. In Section 4.4.1, we compute the distance between hand-designed reward functions, finding EPIC to be the most reliable. NPEC has substantial approximation error, and ERC sometimes erroneously assigns high distance to equivalent rewards. Finally, in Section 4.4.2 we show that EPIC is robust to the exact choice of coverage distribution $\mathcal{D}$, whereas ERC and especially NPEC are highly sensitive to the choice of $\mathcal{D}$.

4.4.1 Comparing hand-designed reward functions

We compare procedurally specified reward functions in four tasks, finding that EPIC is more reliable than the baselines NPEC and ERC, and more computationally efficient than NPEC. Figure 4.2 presents results in the proof-of-concept `PointMass` task. The results for `Gridworld`, `HalfCheetah`, and `Hopper`, in Section B.2.4, are qualitatively similar.

In `PointMass`, the agent can accelerate $\ddot{x}$ left or right on a line. The reward functions include (🐌) or exclude (🐆) a quadratic penalty $\ddot{x}^2$. The sparse reward (`S`) gives a reward of 1 in the region ±0.05 from the origin. The dense reward (`D`) is a shaped version of the sparse reward. The magnitude reward (`M`) is the negative distance of the agent from the origin.



(a) EPIC (b) NPEC (c) ERC

Figure 4.2: Approximate distances between hand-designed reward functions in `PointMass`, where the agent moves on a line trying to reach the origin. EPIC correctly assigns 0 distance between equivalent rewards such as (`D`🐆, `S`🐆) while D_{NPEC}(`D`🐆, `S`🐆) $= 0.58$ and D_{ERC}(`D`🐆, `S`🐆) $= 0.56$. The coverage distribution $\mathcal{D}$ is sampled from rollouts of a policy π_{uni} taking actions uniformly at random. **Key**: The agent has position $x \in \mathbb{R}$, velocity $\dot{x} \in \mathbb{R}$, and can accelerate $\ddot{x} \in \mathbb{R}$, producing future position $x' \in \mathbb{R}$. 🐌 quadratic penalty on control $\ddot{x}^2$, 🐆 no control penalty. `S` is $\text{Sparse}(x) = \mathbb{1}[|x| < 0.05]$, `D` is shaped $\text{Dense}(x, x') = \text{Sparse}(x) + |x'| - |x|$, while `M` is $\text{Magnitude}(x) = -|x|$.

We find that EPIC correctly identifies the equivalent reward pairs (`S`🐌-`D`🐌 and `S`🐆-`D`🐆) with estimated distance $< 1 \times 10^{-3}$. By contrast, NPEC has substantial approximation error: D_{NPEC}(`D`🐆, `S`🐆) $= 0.58$. Similarly, D_{ERC}(`D`🐆, `S`🐆) $= 0.56$ due to ERC's erroneous handling of stochastic initial states. Moreover, NPEC is computationally inefficient: Figure 4.2(b) took 31 hours to compute. By contrast, the figures for EPIC and ERC were generated in less than two hours, and a lower precision approximation of EPIC finishes in just 17 seconds (see Section B.2.6).

4.4.2 Sensitivity of reward distance to coverage distribution

Reward distances should be robust to the choice of coverage distribution $\mathcal{D}$. In Table 4.2 (center), we report distances from the ground-truth reward (`GT`) to reward functions (rows) across coverage distributions $\mathcal{D} \in \{\pi_{\text{uni}}, \pi^*, \texttt{Mix}\}$ (columns). We find EPIC is fairly robust to the choice of $\mathcal{D}$ with a similar ratio between rows in each column $\mathcal{D}$. By contrast, ERC and especially NPEC are substantially more sensitive to the choice of $\mathcal{D}$.

We evaluate in the `PointMaze` MuJoCo task from Fu, Luo, and Levine [49], where a point mass agent must navigate around a wall to reach a goal. The coverage distributions $\mathcal{D}$ are induced by rollouts from three different policies: π_{uni} takes actions uniformly at random, producing broad support over transitions; π^* is an expert policy, yielding a distribution concentrated around the goal; and `Mix` is a mixture of the two. In EPIC, $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{A}}$ are marginalized from $\mathcal{D}$ and so also vary with $\mathcal{D}$.

We evaluate four reward learning algorithms: `Regression` onto reward labels [*target*

Table 4.2: Low reward distance from the ground-truth (GT) in PointMaze-Train predicts high policy return even in unseen task PointMaze-Test. EPIC distance is robust to the choice of coverage distribution $\mathcal{D}$, with similar values across columns, while ERC and especially NPEC are sensitive to $\mathcal{D}$. The table shows approximate distances (1000× scale) of reward functions from GT. The coverage distribution $\mathcal{D}$ is computed from rollouts in PointMaze-Train of a uniform random policy $\boldsymbol{\pi_{\mathrm{uni}}}$, an expert $\boldsymbol{\pi^*}$, and a Mixture of these policies. $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{A}$ are computed by marginalizing $\mathcal{D}$. **Confidence Intervals**: see Table B.7.

Reward	**1000 × $\boldsymbol{D_{\mathrm{EPIC}}}$**			**1000 × $\boldsymbol{D_{\mathrm{NPEC}}}$**			**1000 × $\boldsymbol{D_{\mathrm{ERC}}}$**		
Function	$\boldsymbol{\pi_{\mathrm{uni}}}$	$\boldsymbol{\pi^*}$	Mix	$\boldsymbol{\pi_{\mathrm{uni}}}$	$\boldsymbol{\pi^*}$	Mix	$\boldsymbol{\pi_{\mathrm{uni}}}$	$\boldsymbol{\pi^*}$	Mix
GT	0.06	0.05	0.04	0.04	3.17	0.01	0.00	0.00	0.00
Regress	35.8	33.7	26.1	1.42	38.9	0.35	9.99	90.7	2.43
Pref	68.7	100	56.8	8.51	1333	9.74	24.9	360	19.6
AIRL SO	572	520	404	817	2706	488	549	523	240
AIRL SA	776	930	894	1067	2040	1039	803	722	964
Mirage	17.0	0.05	397	0.68	6.30	597	35.3	<0.01	166

method from 33, section 3.3], Preference comparisons on trajectories [33], and adversarial IRL with a state-only (AIRL SO) and state-action (AIRL SA) reward model [49]. All models are trained using synthetic data from an oracle with access to the ground-truth; see Section B.2.2 for details.

We find EPIC is robust to varying $\mathcal{D}$ when comparing the learned reward models: the distance varies by less than 2×, and the ranking between the reward models is the same across coverage distributions. By contrast, NPEC is highly sensitive to $\mathcal{D}$: the ratio of AIRL SO (817) to Pref (8.51) is 96 : 1 under π_{uni} but only 2 : 1 (2706 : 1333) under π^*. ERC lies somewhere in the middle: the ratio is 22 : 1 (549 : 24.9) under π_{uni} and 3 : 2 (523 : 360) under π^*.

We evaluate the effect of pathological choices of coverage distribution $\mathcal{D}$ in Table B.8. For example, **Ind** independently samples states and next states, giving physically impossible transitions, while **Jail** constrains rollouts to a tiny region excluding the goal. We find that the ranking of EPIC changes in only one distribution, while the ranking of NPEC changes in two cases and ERC changes in all cases.

However, we do find that EPIC is sensitive to $\mathcal{D}$ on Mirage, a reward function we explicitly designed to break these methods. Mirage assigns a larger reward when close to a "mirage" state than when at the true goal, but is identical to GT at all other points. The "mirage" state is rarely visited by random exploration π_{uni} as it is far away and on the opposite side of the wall from the agent. The expert policy π^* is even less likely to visit it, as it is not on or close to the optimal path to the goal. As a result, the EPIC distance from Mirage to GT (Table 4.2, bottom row) is small under π_{uni} and π^*.

In general, any black-box method for assessing reward models — including the rollout method — only has predictive power on transitions visited during testing. Fortunately, we can achieve a broad support over states with `Mix`: it often navigates around the wall due to π^*, but strays from the goal thanks to π_{uni}. As a result, EPIC under `Mix` correctly infers that `Mirage` is far from the ground-truth `GT`.

These empirical results support our theoretically inspired recommendation from Section 4.2: "in general, it is best to choose $\mathcal{D}$ to have broad coverage over plausible transitions." Distributions such as π^* are too narrow, assigning coverage only on a direct path from the initial state to the goal. Very broad distributions such as **Ind** waste probability mass on impossible transitions like teleporting. Distributions like `Mix` strike the right balance between these extremes.

4.5 Conclusion

Our novel EPIC distance compares reward functions directly, without training a policy. We have proved it is a pseudometric, is bounded, and is invariant to equivalent rewards. Empirically, we find EPIC correctly infers zero distance between equivalent reward functions that the NPEC and ERC baselines wrongly consider dissimilar.

Although EPIC can in principle be used to compare arbitrary reward models, we have only evaluated on reward models trained on synthetic data in continuous control tasks. An important direction for future work is to apply EPIC to models trained on real-world data in a broader variety of domains, such as image-based tasks. Such models will have a higher EPIC distance from the ground-truth than models trained on synthetic data. However, some algorithms may be more robust to imperfect feedback than others, potentially changing our ranking of algorithms implied from Table 4.2.

Standardized metrics are an important driver of progress in machine learning. Unfortunately, traditional policy-based metrics do not provide any guarantees as to the fidelity of the learned reward function. We believe the EPIC distance will be a highly informative addition to the evaluation toolbox, and would encourage researchers to report EPIC distance in addition to policy-based metrics. Our implementation of EPIC and our baselines, including a tutorial and documentation, are available at `https://github.com/HumanCompatibleAI/evaluating-rewards`.

Chapter 5

Distance metrics predict regret

In the previous chapter, we introduced the EPIC distance. In this chapter, we show that the EPIC distance can be used to bound the regret of optimal policies even under different transition dynamics. Moreover, we confirm empirically that it predicts policy training success, even in situations which do not satisfy the assumptions made in our regret bound theorem.

5.1 A regret bound for EPIC

Theorem 5.1.1. *Let M be a γ-discounted MDP$\backslash R$ with finite state and action spaces $\mathcal{S}$ and $\mathcal{A}$. Let $R_A, R_B : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be rewards, and π_A^*, π_B^* be respective optimal policies. Let $\mathcal{D}_\pi(t, s_t, a_t, s_{t+1})$ denote the distribution over transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ induced by policy π at time t, $\mathcal{D}(s, a, s')$ be the coverage distribution used to compute D_{EPIC}, and $\mathcal{D}_\mathcal{S}(s), \mathcal{D}_\mathcal{A}(a)$ be the distributions defining the canonicalization in D_{EPIC}. Assume the coverage distribution is set equal to the canonicalization distribution: $\mathcal{D}(s, a, s') = \mathcal{D}_\mathcal{S}(s)\mathcal{D}_\mathcal{A}(a)\mathcal{D}_\mathcal{S}(s') \,\forall\, s, s' \in \mathcal{S}, a \in \mathcal{A}$. Suppose there exists $K > 0$ such that $K\mathcal{D}(s_t, a_t, s_{t+1}) \geqslant \mathcal{D}_\pi(t, s_t, a_t, s_{t+1})$ for all times $t \in \mathbb{N}$, triples $(s_t, a_t, s_{t+1}) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, and policies $\pi \in \{\pi_A^*, \pi_B^*\}$. Then, the regret under R_A from executing π_B^* instead of π_A^* is at most*

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant 16K\|R_A\|_2 \, (1-\gamma)^{-1} \, D_{\text{EPIC}}(R_A, R_B),$$

where $G_R(\pi)$ is the return of policy π under reward R, and the L^2 norm $||R_A||_2$ is taken with respect to the coverage distribution $\mathcal{D}$.

We generalize the regret bound to continuous spaces in theorem B.5.4 via a Lipschitz assumption and with Wasserstein distance replacing K. Importantly, the returns of π_A^* and π_B^* **converge** as $D_{\text{EPIC}}(R_A, R_B) \to 0$ in both cases, no matter which reward function is used for evaluation.

The theorem supplies a non-trivial bound when the coverage distribution $\mathcal{D}$ has adequate support for transitions occurring in rollouts of π_A^* and π_B^*. The bound is tightest when $\mathcal{D}$ is similar to $\mathcal{D}_{\pi_A^*}$ and $\mathcal{D}_{\pi_B^*}$. However, computing π_A^* and π_B^* is often intractable. The

MDP M may be unknown, such as when making predictions about an unseen deployment environment. Even when M is known, RL is computationally expensive and may fail to converge in non-trivial environments.

In finite cases, a uniform $\mathcal{D}$ satisfies the requirements with $K \leqslant |\mathcal{S}|^2|\mathcal{A}|$. In general, it is best to choose $\mathcal{D}$ to have broad coverage over plausible transitions. Broad coverage ensures adequate support for $\mathcal{D}_{\pi_A^*}$ and $\mathcal{D}_{\pi_B^*}$. But excluding transitions that are unlikely or impossible to occur leads to tighter regret bounds due to a smaller K (finite case) or Wasserstein distance (continuous case).

The theorem requires the coverage and canonicalization distributions to be equal. However, in practice it can be desirable to set the coverage distribution to something narrower. For example, setting $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{A}$ to be uniform over $\mathcal{S}$ and $\mathcal{A}$ means the canonicalization distribution $\mathcal{D}_\mathcal{S} \times \mathcal{D}_\mathcal{A} \times \mathcal{D}_\mathcal{S}$ places equal weight on all transitions, including many impossible transitions (e.g., teleportation in gridworlds). Putting more weight on transitions that are likely to happen and placing no support on transitions we know *a priori* to be impossible seem likely to make the EPIC distance a better predictor of reward function similarity in practice. Empirically, the EPIC distance remains highly correlated to regret even when the coverage and canonicalization distributions do not match (demonstrated in Section 5.2).

While EPIC upper bounds policy regret, it does not lower bound it. In fact, no reward distance can lower bound regret in arbitrary environments. For example, suppose the deployment environment transitions to a randomly chosen state independent of the action taken. In this case, all policies obtain the same expected return, so the policy regret is always zero, regardless of the reward functions.

5.2 Experiments

We find that low distance from the ground-truth reward `GT` (Table 5.1, center) predicts high `GT` return (Table 5.1, right) of policies optimized for that reward. Moreover, the distance is predictive of return not just in `PointMaze-Train` where the reward functions were trained and evaluated in, but also in the unseen variant `PointMaze-Test`. This is despite the two variants differing in the position of the wall such that policies for `PointMaze-Train` run directly into the wall in `PointMaze-Test`. Notably, the regret bound (Theorem 5.1.1) does not apply to this setting because the canonicalization distribution does not match the coverage distribution. However, it seems in practice that the EPIC distance remains highly predictive of regret even when this assumption is violated.

Both `Regress` and `Pref` achieve very low distances at convergence, producing near-expert policy performance. The `AIRL SO` and `AIRL SA` models have reward distances an order of magnitude higher and have poor policy performance. Yet, intriguingly, the *generator* policies for `AIRL SO` and `AIRL SA` — trained simultaneously with the reward — perform reasonably in `PointMaze-Train`, achieving -5.43 and -5.05 respectively. This suggests the learned rewards are reasonable on the subset of transitions taken by the generator policy yet fail to transfer to the different transitions taken by a policy being trained from scratch.

Table 5.1: Low reward distance from the ground-truth (`GT`) in `PointMaze-Train` predicts high policy return even in unseen task `PointMaze-Test`. EPIC distance is robust to the choice of coverage distribution $\mathcal{D}$, with similar values across columns, while ERC and especially NPEC are sensitive to $\mathcal{D}$. **Center**: approximate distances (1000× scale) of reward functions from `GT`. The coverage distribution $\mathcal{D}$ is computed from rollouts in `PointMaze-Train` of a uniform random policy $\boldsymbol{\pi}_{\mathbf{uni}}$, an expert $\boldsymbol{\pi}^*$, and a `Mix`ture of these policies. $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{A}$ are computed by marginalizing $\mathcal{D}$. **Right**: mean `GT` return over 9 seeds of RL training on the reward in `PointMaze-{Train,Test}`. **Confidence Intervals**: see Table B.7.

Reward	**$1000 \times D_{\mathrm{EPIC}}$**			**$1000 \times D_{\mathrm{NPEC}}$**			**$1000 \times D_{\mathrm{ERC}}$**			**Return**	
Function	$\boldsymbol{\pi}_{\mathbf{uni}}$	$\boldsymbol{\pi}^*$	`Mix`	$\boldsymbol{\pi}_{\mathbf{uni}}$	$\boldsymbol{\pi}^*$	`Mix`	$\boldsymbol{\pi}_{\mathbf{uni}}$	$\boldsymbol{\pi}^*$	`Mix`	**Train**	**Test**
`GT`	0.06	0.05	0.04	0.04	3.17	0.01	0.00	0.00	0.00	−5.19	−6.59
`Regress`	35.8	33.7	26.1	1.42	38.9	0.35	9.99	90.7	2.43	−5.47	−6.30
`Pref`	68.7	100	56.8	8.51	1333	9.74	24.9	360	19.6	−5.57	−5.04
`AIRL SO`	572	520	404	817	2706	488	549	523	240	−27.3	−22.7
`AIRL SA`	776	930	894	1067	2040	1039	803	722	964	−30.7	−29.0
`Mirage`	17.0	0.05	397	0.68	6.30	597	35.3	<0.01	166	−30.4	−29.1

Figure B.6 shows reward distance and policy regret during reward model training. The lines all closely track each other, showing that the distance to `GT` is highly correlated with policy regret for intermediate reward checkpoints as well as at convergence. `Regress` and `Pref` converge quickly to low distance and low regret, while `AIRL SO` and `AIRL SA` are slower and more unstable.

5.3 Conclusion

We have shown that EPIC bounds the regret of optimal policies (Theorems 4.2.8, 4.2.9, and 5.1.1). This bound assumes the coverage distribution is a product of state and action marginals. However, we find empirically that even when this assumption is violated, the EPIC distance of learned reward functions to the ground-truth reward still predicts the return of policies optimized for the learned reward. Notably, this prediction holds even when the distance is computed in `PointMaze-Train` and the regret is computed for the unseen `PointMaze-Test` environments.

This is important since it is common for the initial state distribution or transition dynamics to change between the training environment where the reward function was learned and the test environment where the system is deployed [130, 8, 121]. For example, one might learn a reward and policy in simulation and then fine-tune the policy in the real world with the learned reward. Moreover, in life-long learning the policy is continually trained with the learned reward. In this case, the reward must also be robust to unintentional changes that may occur over time in the deployment environment. Although we can only test with intentional changes, we expect our method to also be predictive of robustness.

Chapter 6

Understanding learned reward functions

It is often possible to learn a reward function that would be difficult or impossible to manually specify. However, learned rewards may fail to represent user preferences, so it is important to be able to validate the learned reward function prior to deployment. One promising approach is to apply interpretability tools to the reward function to spot potential deviations from the user's intention. Existing work has applied general-purpose interpretability tools to understand learned reward functions. We propose exploiting the intrinsic structure of reward functions by first *preprocessing* them into simpler but equivalent reward functions, which are then visualized. We introduce a general framework for such reward preprocessing and propose concrete preprocessing algorithms. Our empirical evaluation shows that preprocessed rewards are often significantly easier to understand than the original reward.

6.1 Introduction

Manually specifying reward functions has many challenges, but it does at least have the advantage that we can easily understand the reward function (if not the consequences of optimizing it) by reading the implementation. By contrast, learned reward functions are often black-box models. This makes it challenging to evaluate a learned reward function in order to spot potential failure modes prior to deployment.

In Chapter 4, we introduced the EPIC distance to quantify the difference between two functions. This works great when a ground-truth reward is available, as we can evaluate a learned reward simply by computing its EPIC distance to the ground-truth. However, reward learning is most useful precisely when we do *not* have access to a ground-truth reward. In this setting, EPIC may still have some limited utility for comparing and perhaps clustering several learned reward models, but it cannot tell us if any of these learned models are correct.

In Michaud, Gleave, and Russell [105] we instead suggest *interpreting* reward models to verify they capture user preferences. We find that existing interpretability methods such as saliency maps can help understand reward models. However, we also find significant limitations in this approach, concluding that "reward interpretability may need significantly different methods from policy interpretability."

We believe that significant advances in reward interpretability can be made by taking advantage of the special structure of reward functions. In particular, many different reward functions are *equivalent*, in the sense that they induce the same optimal policies—no matter the environment dynamics. Given a learned reward model, we can apply transformations that do not change the optimal policy, but simplify the reward function. We can then visualize this simplified reward instead of the original. We call this approach *reward preprocessing*, as we "preprocess" the reward model prior to visualization.

Our framework for reward preprocessing involves two key components: 1) a class of reward transformations that yield equivalent reward functions in some sense (e.g., by preserving the optimal policy under arbitrary environment dynamics), and 2) an objective that measures how interpretable a given reward function is. We then optimize over the class of transformations using the given objective to find the most interpretable equivalent reward function. A key property of this framework is that the learned reward model is treated as a black box. This means that it may use an arbitrary function approximator and can be learned using any reward learning algorithm and feedback modality.

In summary, our key contributions are: 1) a novel framework that exploits the intrinsic structure of reward functions to increase their interpretability before visualization; 2) two concrete applications of this framework, using different objectives for interpretability; and 3) an empirical evaluation, finding that our methods often significantly improve interpretability.

6.1.1 Related work

Interpreting reward models has recently begun to receive some attention. Russell and Santos [137] apply standard interpretability methods from supervised learning to reward functions. Specifically, they use feature importance estimates from a simple fitted global model and from LIME [134] to interpret the reward function.

Globally fitting a simpler model to a reward function has some similarities to our reward preprocessing approach. However, a major difference is that the simple model will usually not be equivalent to the original reward function. In contrast, we learn an *equivalent* but still simplified reward model. This is possible because we exploit the structure that reward functions naturally have, whereas Russell and Santos [137] only apply preexisting interpretability methods.

Michaud, Gleave, and Russell [105] also apply existing interpretability methods to understand reward models. In contrast to Russell and Santos, they work directly with the given reward, without fitting a simpler model. They suggest and combine three different approaches, namely gradient saliency maps, occlusion maps, and handcrafted counterfactual inputs. All of these methods can also be applied to supervised learning more broadly and do not take advantage of the structure of reward functions.

Our reward preprocessing framework is complementary to these methods for interpreting reward functions. We advocate first preprocessing a given reward to select a maximally comprehensible equivalent reward function. The resulting reward function can then be visualized or otherwise interpreted using a range of techniques.

While there is only a handful of work seeking to understand learned reward functions, considerably more work has focused on interpreting *policies* [129]. One approach is to learn a policy from a class of intrinsically simple functions rather than neural networks [167]. Alternatively, Juozapaitis et al. [76] present a method that explains policy actions by an additive decomposition of Q-values. Another promising recent direction is using causal models to explain policy behavior [99, 39].

Devidze et al. [41] approach interpretability of reward functions from a different angle: rather than interpreting a complex *learned* reward function, they aim to *design* a reward function that trades off between interpretability (operationalized as sparsity) and ease of policy optimization.

6.2 The reward preprocessing framework

Our interpretability method operates on a reward function $r(s, a, s')$, where s is the current state, a is the action taken in that state, and s' is the next state. Our method only requires the ability to evaluate r: there are no restrictions on how r is computed or how it was learned. From r, we produce a simpler but equivalent reward function r', which we then visualize.

Concrete instantiations of this framework must make two choices. First, they must specify which reward functions are deemed equivalent via an equivalence relation $\sim$. Second, they must provide some measure of "simplicity" or "interpretability," represented by a cost function J. We then seek to find a minimum cost reward function r' that is equivalent to r:

$$r' := \arg\min_{\hat{r} \sim r} J(\hat{r}). \tag{6.1}$$

In the following, we discuss how to choose the equivalence relation $\sim$ and cost function J.

6.2.1 Equivalence relation

We would like to treat two rewards as equivalent if they will produce the same behavior in the intended downstream application. It is known that potential shaping [111] and rescaling by a positive constant never change the ordering of policies (see Section 2.2). It is therefore safe to treat such rewards as equivalent for most applications.

However, some applications permit a broader notion of equivalence. For example, if the reward model will only ever be used for policy optimization in a specific task, then we can include any transformations that preserve optimal policies in that task. A simple example is S'-redistribution: moving reward between different successor states, while preserving $\mathbb{E}_{S'} r(s, a, S') = \mathbb{E}_{S'} r'(s, a, S')$. This will not change the optimal policy, so long as the transition dynamics determining S' remain fixed. Skalse et al. [155] characterize a variety of such equivalence classes under varying assumptions.

6.2.2 Choosing cost functions

The cost function J should represent the interpretability of a reward function. Of course, no simple objective can capture the entire concept of interpretability since reward functions may be interpretable for a variety of reasons. For example, *sparse* rewards are often interpretable, as the user can pay attention only to the few transitions on which the agent receives a non-zero reward. However, a dense reward could still be easy to understand if it has some other simple structure: for example, taking on only two different values depending on which region of the world the agent is in.

Instead of looking for a single cost function that completely characterizes interpretability, we therefore suggest using *multiple* cost functions, each of which describes some condition that is *sufficient* but not *necessary* for interpretability. We can then find an optimal equivalent reward for each of the cost functions and present all of these rewards for the user to choose between. Provided the cost functions are on a comparable scale, we can also rank the reward functions, presenting the lowest-cost rewards first.

Another factor determining the appropriate cost functions is the method used for visualization. For example, a reward function that has sparse output is ideal if we wish to show the user the reward of particular transitions. However, we might prefer sparsity in the *features* that the reward depends on if we are using higher-level visualization methods like saliency maps.

6.3 Methodology

In this section, we describe a few simple concrete instances of our reward preprocessing framework. Despite their simplicity, we find in Section 6.4 that they nonetheless can yield significant improvements. However, these choices are likely far from optimal and so should be viewed as establishing a lower bound on the benefit obtainable from reward preprocessing.

6.3.1 Potential-shaping equivalence relation

We define two rewards to be equivalent, $r \sim r'$, if they are equal up to *potential shaping* [111]. Specifically, $r \sim r'$ if there exists some real-valued state-only function Φ called a *potential* for which $r'(s, a, s') = r(s, a, s') + \gamma\Phi(s') - \Phi(s)$, where $\gamma \in [0, 1)$. Potential shaping changes the returns of an episode by only the potential $\Phi(s_0)$ of the initial state (in the finite horizon case, the potential of terminal states is restricted to be zero). Since the policy does not affect the initial state, the ordering over policies is invariant under potential shaping. This holds for *arbitrary* transition dynamics and initial state distributions. Therefore, rewards related to each other by potential shaping can be considered equivalent even under transfer to different environment dynamics.

A notable advantage of potential shaping for our purposes is that it is very easy to optimize over the resulting equivalence class. We simply parameterize the potential as a neural network $\Phi_\theta(s)$ with parameters θ. Then, the optimization problem from Equation (6.1) becomes

$$\arg\min_\theta J(r'_\theta)\,,\ \text{where}\, r'_\theta(s,a,s') = r(s,a,s') + \gamma\Phi_\theta(s') - \Phi_\theta(s). \tag{6.2}$$

We optimize Equation (6.2) using (stochastic) gradient descent. This requires a differentiable cost function J but does *not* require the reward function r to be differentiable.

Rewards differing by a positive scale factor also produce the same policy ordering. However, since our visualization techniques can handle rewards at a range of scales, we choose to preserve the scale during preprocessing. Accordingly, we do not include rescaled rewards as equivalent.

6.3.2 Cost functions

We evaluate two types of cost functions: a sparsity-inducing one based on the L^1 norm and a smoothness-inducing measure of absolute deviation. In tabular (gridworld) settings, we evaluate these cost functions on a uniform distribution $\mathcal{D}$ over all possible transitions. In continuous control environments, we evaluate on transitions sampled from the same distribution $\mathcal{D}$ used for visualization.

Sparse rewards are easy to understand as the user only needs to attend to rewards with non-zero transitions. However, the L^0 norm is non-differentiable. Moreover, even if the ground-truth reward is sparse, learned reward functions are usually not exactly equivalent to a sparse reward due to the presence of noise. We therefore use two different relaxed notions of sparsity: the L^1 norm $|r|$ and the slightly transformed version $\log(1 + |r|)$. In particular, we minimize

$$J_{\text{sparse}}(r) := \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\, f\big(r(s,a,s')\big)\,, \tag{6.3}$$

where $\mathcal{D}$ is the distribution over transitions and $f(x)$ is either $|x|$ or $\log(1 + |x|)$.

An alternative is to minimize the fluctuations between rewards of transitions adjacent in time. This creates a smoothly varying reward signal. The user can then understand the reward by looking at the trend over time. This frees the user from having to attend to the reward at every single transition, similar to sparsity. Again, we use an L^1 and a logarithmic version of such a smoothness cost:

$$J_{\text{smooth}}(r) := \mathbb{E}_{(s_t,a_t,s_{t+1},a_{t+1},s_{t+2})\sim\mathcal{D}}\, f\big(r(s_t,a_t,s_{t+1}) - r(s_{t+1},a_{t+1},s_{t+2})\big)\,. \tag{6.4}$$

6.4 Results

We evaluate our methods in two environments: a gridworld, with varying rewards, and the classic mountain car continuous control task [24]. We test our method with a mixture of hand-designed and learned rewards. The hand-designed rewards consist of a simple ground-truth reward, with shaping and/or noise added to challenge the preprocessing method. The learned rewards are trained via either adversarial inverse reinforcement learning [49, AIRL], or deep reinforcement learning from human preferences [33, DRLHP]. Both methods are trained on synthetic data, consisting of rollouts from an expert policy (AIRL) or preference comparisons induced by the ground-truth reward (DRLHP).

In gridworld experiments, we use a tabular potential and reward model. That is, we learn a separate value $\Phi(s)$ and $r(s, a, s')$ for each state and transition. In mountain car, we use small MLPs for the reward model and potentials, except for some cases where a linear potential is sufficient to find a simple equivalent reward. Our code is available at `https://github.com/HumanCompatibleAI/reward-preprocessing`.

6.4.1 Simplifying shaped rewards

We start by testing our method in a gridworld setting [186]. While unrealistic, gridworlds have the considerable benefit of allowing the entire reward function to be easily visualized. This therefore allows a more thorough evaluation of our method than in other tasks.

In Figures 6.1 and 6.2, we visualize gridworld rewards before (leftmost column) and after (middle and right column) our preprocessing methods. In the `Goal` environment in Figure 6.1, the reward is simply 1 on a single goal square in the top right corner and 0 everywhere else. This is readily understood and our preprocessing largely retains this reward unchanged. However, when we add shaping with the Manhattan distance from the goal (second row), the reward becomes much harder to understand. Our preprocessing, however, is able to simplify this shaped reward to something close to the original sparse objective. Similar results hold for the negative Manhattan distance from the goal (third row) and the particularly confusing random shaping (last row).

In the `Path` environment in Figure 6.2, the original reward (top left) prefers a specific path for reaching the goal state. Once again, the shaped versions obscure this, but preprocessing reliably recovers a simple and interpretable reward.

In these plots, we use the L^1 version of the sparsity cost and the logarithmic version of the smoothness cost. These work slightly better than the other versions, but the difference is very small. The results for all versions can be found in Figures C.1 to C.4 in the appendix.

6.4.2 Understanding learned rewards

In the previous experiment, all the reward functions were exactly equivalent to the simple original reward. By contrast, learned reward models may be noisy or contain systematic errors, and may not be equivalent to any simple reward. To evaluate how our method performs in this more realistic setting, we trained reward models from demonstrations (AIRL) and preference comparisons (DRLHP) on synthetic data in both of the previous `Goal` and `Path` environments. The results of applying our preprocessing method are shown in Figures C.5 to C.12 in the appendix.

For the reward model learned using preference comparisons (DRLHP), even the preprocessed models look very noisy. The goal state does tend to be somewhat more visible in the preprocessed than the unprocessed rewards, but neither are easy to understand. The reward model learned by AIRL differs even more from the ground-truth reward, and potential shaping is unable to bridge that gap.

It might be possible to remove more of the noise by using a larger equivalence class than potential shaping. However, expanding the equivalence class might mean the preprocessed reward would no longer induce the same optimal policy as the unmodified reward in some environment dynamics. Indeed, the fact that potential shaping is not sufficient to remove the noise suggests that what DRLHP and AIRL have learned is not just a complex but validly shaped version of the ground-truth reward.

6.4.3 Mountain car

Since the mountain car environment has an infinite number of possible transitions, we cannot plot the rewards of all possible transitions as we did in the gridworld tasks. Instead, we visualize reward functions by plotting the reward signal over time during expert trajectories. Figure 6.3 visualizes two learned reward models (left) and the reward signal after preprocessing with a log sparse (middle) and log smooth (right) cost function.

The model in the top row was trained using DRLHP on synthetically generated preferences. Specifically, we sampled Boltzmann-rational preferences between trajectory fragments based on the ground-truth reward. In the second row, we first learned an optimal state value function for the mountain car environment and then used this to shape the ground-truth reward before generating preferences. This simulates human feedback, which may be shaped compared to a sparse ground-truth since humans already reward incremental progress [33].

As in the gridworld setting, both the learned and preprocessed rewards are noisy. However, the preprocessed reward functions are still significantly simpler than the learned models, especially in the shaped case. The sparsity cost function performs better here than the smoothness cost. Figure 6.3 uses the logarithmic version of both but the L^1 version in Figure C.15 yields almost exactly the same results.

Notably, the residual noise after preprocessing in Figure 6.3 is likely not removable by potential shaping. In particular, we find in Figures C.13 and C.14 that preprocessing on shaped versions of the ground-truth reward recover simple, noise-free rewards. The residual noise is therefore likely an accurate depiction of errors in the learned reward.

6.5 Limitations and future work

One limitation of our approach is that while potential shaping does not change the optimal policy, it can make the policy optimization problem easier or harder. Consequently, the policy learned by an RL algorithm might well differ between the unmodified learned reward and the theoretically “equivalent” reward used for visualization. This issue is most significant in environments where policy optimization can be challenging. Reasoning about how shaping affects RL algorithm performance is challenging, so this is only a significant factor when the tool is being used by trained practitioners.

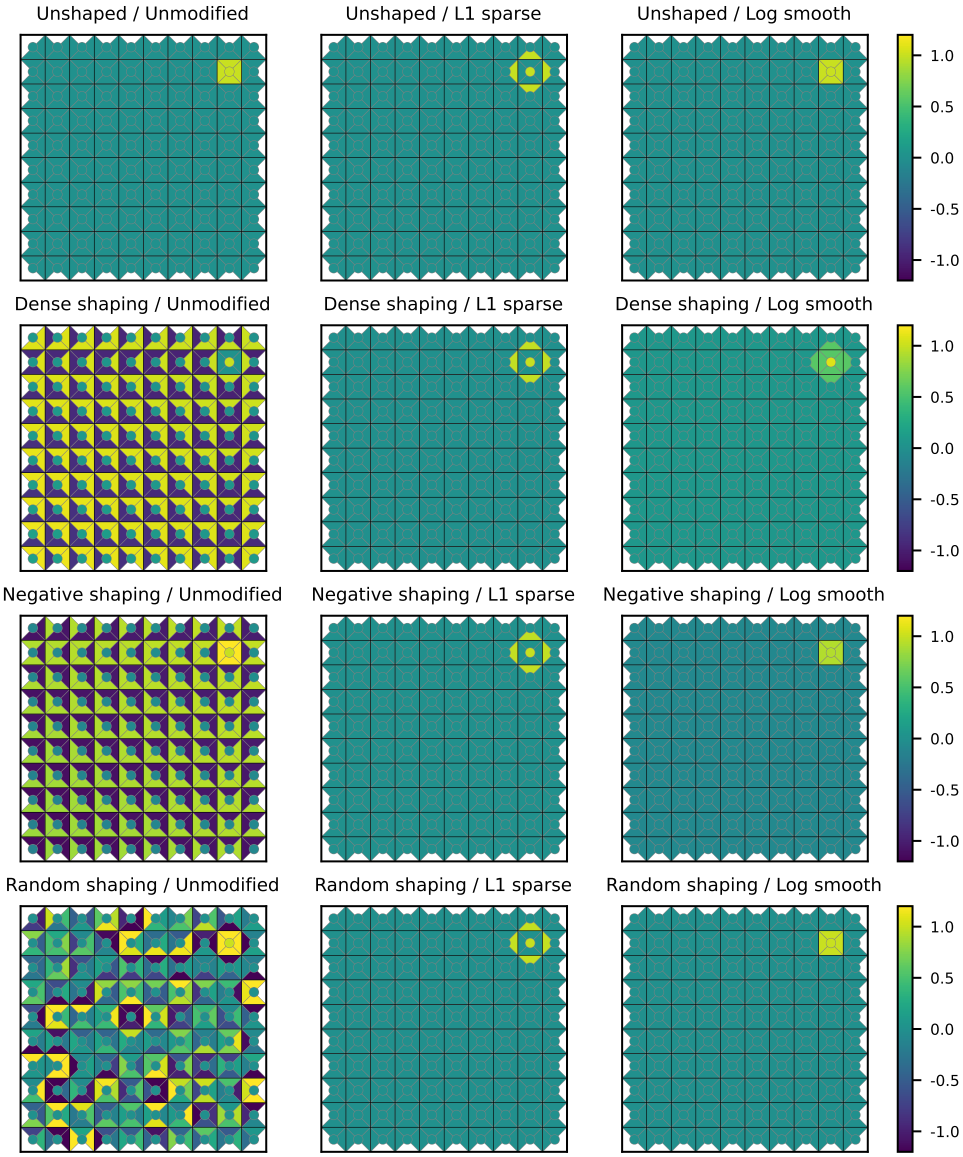


Figure 6.1: Preprocessing can recover a sparse reward from complex shaping. The original sparse `Goal` reward is shown in the top left, with three shaped versions below. These rewards are shown after preprocessing with the sparsity (middle) and smoothness (right) cost functions. The preprocessed rewards are easy to understand, and are similar across a range of shaping. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

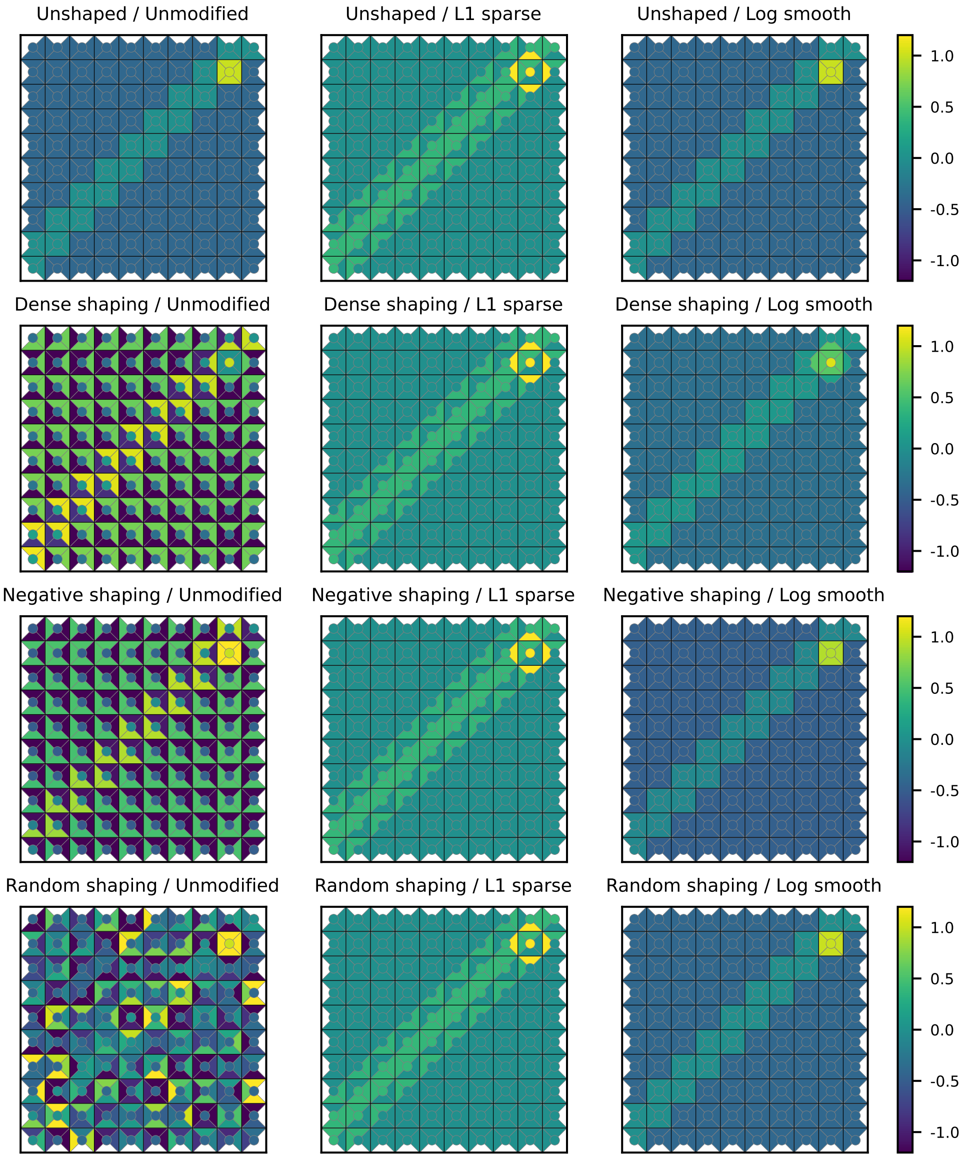


Figure 6.2: Preprocessing recovers a simpler dense reward from three complex shaped rewards. The original `Path` reward (top-left) incentivizes following a diagonal path to the goal state. The shaped versions below obscure this pattern, but preprocessing recovers something similar to the original reward. This is notable as the original reward is not sparse, so has a relatively high cost under the L^1 norm, but is still lower cost than the highly complex shaped rewards. Each heatmap shows the rewards for all possible transitions in a 10×10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

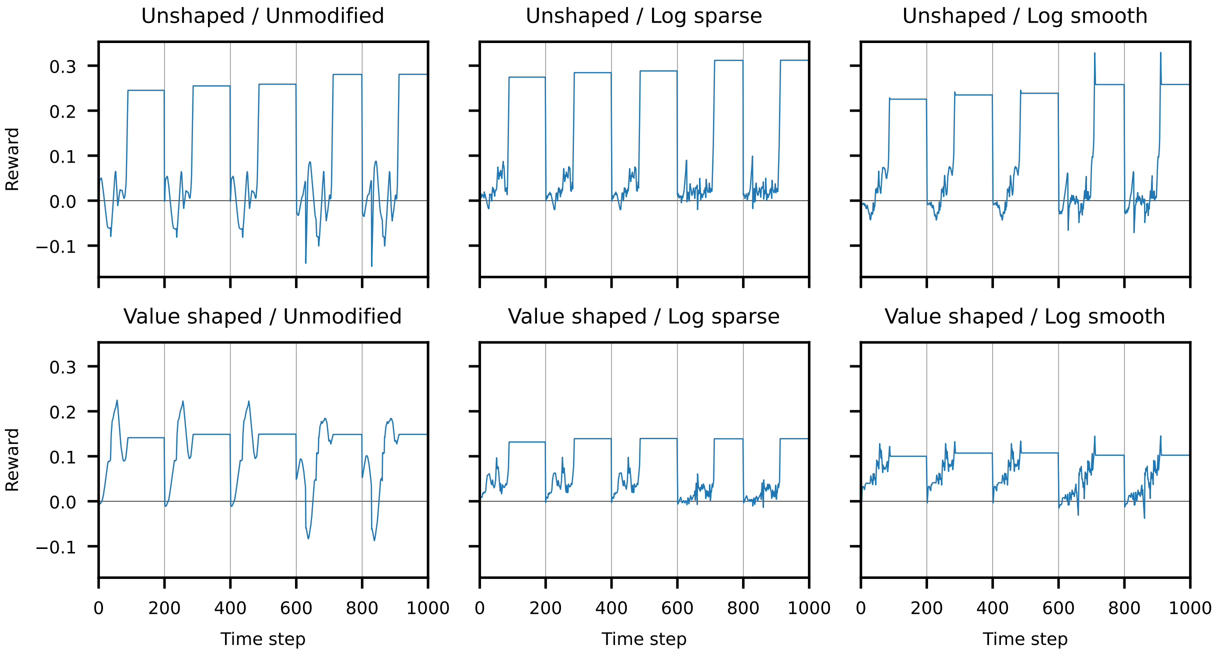


Figure 6.3: Preprocessing can simplify complex learned reward models for mountain car. The left column shows reward models learned using synthetic preference comparisons based on the ground-truth reward (top), and the ground-truth shaped with an optimal value function (bottom). Preprocessing for sparsity (middle) and smoothness (right) produces simpler and less noisy reward curves, especially in the shaped setting. Each plot shows the reward during a rollout over five episodes (separated by the gray vertical lines).

The above limitation is a way in which potential shaping can be too *big* an equivalence class. However, there is also a sense in which it is too *small*. In practice, we do not usually care about a reward function transferring to *all* possible transition dynamics. If it is known the transition dynamics satisfy certain invariants, then we may be able to use a larger equivalence class while still guaranteeing optimal policy preservation.

In addition to modifying the equivalence class, there are also numerous alternative cost functions that could be employed. In particular, the cost functions we suggest are targeted at the visualizations we use in this paper. Other visualizations might benefit from different cost functions. However, it seems likely that the basic concepts of sparsity and smoothness will be useful in many settings. For example, visualizations using gradient saliency maps might benefit from maximizing the sparsity of the gradients rather than of the rewards themselves.

6.6 Conclusion

We have introduced a novel framework to preprocess reward functions prior to visualization. Our empirical results demonstrate this methodology can recover simple reward functions from shaped versions of ground-truth rewards. Moreover, our method can substantially simplify even noisy learned reward models. However, some low-quality learned reward models are still difficult to understand even with our method, suggesting that reward learning algorithms often converge to models significantly different from the user's intended preferences.

Part II

Agent robustness

Chapter 7

The adversarial policies threat model

Deep reinforcement learning (RL) policies are known to be vulnerable to adversarial perturbations to their observations, similar to adversarial examples for classifiers. However, an attacker is not usually able to directly modify another agent's observations. This might lead one to wonder: is it possible to attack an RL agent simply by choosing an *adversarial policy* acting in a multi-agent environment so as to create *natural* observations that are adversarial? In this chapter, we formalize this threat model and review related work. In Chapter 8, we will demonstrate this attack against state-of-the-art policies controlling simulated humanoid robots. Next, in Chapter 9 we demonstrate a similar attack against professional-level Go-playing AI systems. Finally, in Chapter 10 we introduce and evaluate defenses against this attack.

7.1 Introduction

Most study of adversarial examples has focused on small ℓ_p-norm perturbations to images, which Szegedy et al. [158] found cause misclassifications in a variety of models, even though the changes are visually imperceptible to a human. Most prior work studying adversarial examples in RL has also assumed an ℓ_p-norm threat model. In particular, Huang et al. [67], Kos and Song [83], and Lin et al. [93] showed that deep RL policies are vulnerable to small perturbations in image observations.

RL has been applied in settings as varied as autonomous driving [42], negotiation [91], and automated trading [113]. In domains such as these, an attacker cannot usually directly modify the defender policy's input. For example, in autonomous driving, pedestrians and other drivers can take actions in the world that affect the camera image, but only in a physically realistic fashion. They cannot add noise to arbitrary pixels, or make a building disappear. Similarly, in financial trading, an attacker can send orders to an exchange which will appear in the defender's market data feed, but the attacker cannot modify observations of a third party's orders.

We therefore introduce a physically realistic threat model where the adversary can take actions in a shared environment with the defender but cannot directly perturb the defender's observations. Our work was in part inspired by prior criticisms of the ℓ_p model. For example, Gilmer et al. [52] argue that attackers are not limited to small perturbations and can instead construct new images or search for naturally misclassified images. Similarly, Uesato et al. [165] argue that the ℓ_p model is merely a convenient local approximation for the true worst-case risk. We follow Goodfellow et al. [57] in viewing adversarial examples as any input "that an attacker has intentionally designed to cause the model to make a mistake."

A similar threat model was used in Behzadan and Munir [19], who pit autonomous vehicles against an adversarial car. However, in collaborative games like driving, even the optimal policy may be exploitable; we therefore focus on zero-sum games. Lanctot et al. [86] showed that agents may be tightly coupled to the agents they were trained with, causing seemingly strong polices to fail against new opponents. However, the agents we attack beat a range of opponents, so they are not coupled.

This work follows a rich tradition of worst-case analysis in RL. In robust MDPs, the transition function is chosen adversarially from an *uncertainty set* [10, 159]. Doyle et al. [43] solve the converse problem: finding the set of transition functions for which a policy is optimal. Methods also exist to verify controllers or find a counterexample to a specification. Bastani, Pu, and Solar-Lezama [16] verify decision trees distilled from RL policies, while Ghosh et al. [51] test black-box closed-loop simulations. Ravanbakhsh and Sankaranarayanan [132] can even synthesize controllers robust to adversarial disturbances. Unfortunately, these techniques are only practical in simple environments with low-dimensional adversarial disturbances. By contrast, while our method lacks formal guarantees, it can test policies in complex multi-agent tasks, and it naturally scales with improvements in RL algorithms.

7.2 Framework

We model the defender as playing against an opponent in a two-player Markov game [149]. Our threat model, illustrated in Figure 7.1, assumes the attacker can control the opponent, in which case we call the opponent an adversary. We denote the adversary and defender by subscript α and ν respectively. The game $M = (\mathcal{S}, (\mathcal{A}_\alpha, \mathcal{A}_\nu), T, (R_\alpha, R_\nu))$ consists of state set $\mathcal{S}$, action sets $\mathcal{A}_\alpha$ and $\mathcal{A}_\nu$, and a joint state transition function $T : \mathcal{S} \times \mathcal{A}_\alpha \times \mathcal{A}_\nu \to \Delta(\mathcal{S})$ where $\Delta(\mathcal{S})$ is a probability distribution on $\mathcal{S}$. The reward function $R_i : \mathcal{S} \times \mathcal{A}_\alpha \times \mathcal{A}_\nu \times \mathcal{S} \to \mathbb{R}$ for player $i \in \{\alpha, \nu\}$ depends on the current state, next state, and both players' actions. Each player wishes to maximize their (discounted) sum of rewards.

The adversary is allowed unlimited black-box access to actions sampled from π_v, but is not given any white-box information such as weights or activations. We further assume the defender agent follows a fixed stochastic policy π_v, corresponding to the common case of a pre-trained model deployed with static weights. Note that in safety-critical systems, where attacks like these would be most concerning, it is standard practice to validate a model and then freeze it, so as to ensure that the deployed model does not develop any new issues due

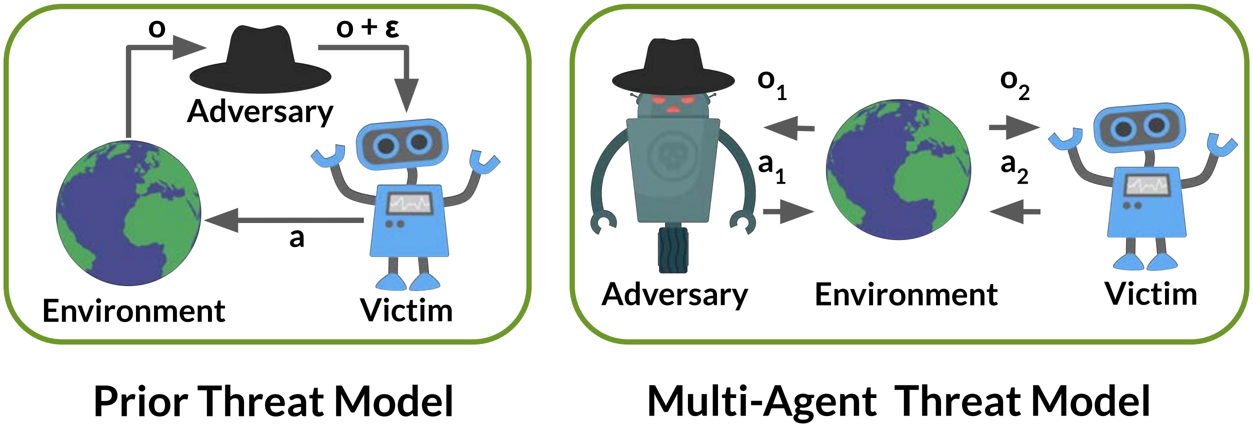


Figure 7.1: **Left**: Prior work models the adversary as being able to perturb arbitrary sensory inputs of the defender, but only by a small amount. **Right**: The adversarial policies threat model treats the adversary as an agent acting in an environment shared with the defender. It has no special powers, and only takes the same (or similar) set of actions as the defender.

to retraining. Therefore, a fixed defender is a realistic reflection of what we might see with RL-trained policies in real-world settings, such as with autonomous vehicles.

Since the defender policy π_ν is held fixed, the two-player Markov game M reduces to a single-player MDP $M_\alpha = (\mathcal{S}, \mathcal{A}_\alpha, T_\alpha, R'_\alpha)$ that the attacker must solve. The state and action space of the adversary are the same as in M, while the transition and reward functions have the defender policy π_ν embedded:

$$T_\alpha\left(s, a_\alpha\right) = T\left(s, a_\alpha, a_\nu\right) \qquad \text{and} \qquad R'_\alpha(s, a_\alpha, s') = R_\alpha(s, a_\alpha, a_\nu, s'),$$

where the defender's action is sampled from the stochastic policy $a_\nu \sim \pi_\nu(\cdot \mid s)$. The attacker's goal is to find an adversarial policy π_α maximizing the sum of discounted rewards:

$$\sum_{t=0}^{\infty} \gamma^t R_\alpha(s^{(t)}, a_\alpha^{(t)}, s^{(t+1)}), \quad \text{where } s^{(t+1)} \sim T_\alpha(s^{(t)}, a_\alpha^{(t)}) \text{ and } a_\alpha \sim \pi_\alpha(\cdot \mid s^{(t)}). \tag{7.1}$$

Note the MDP's dynamics T_α will be unknown even if the Markov game's dynamics T are known, since the defender policy π_ν is a black box. Consequently, the attacker must solve an RL problem.

We measure two primary success metrics: the *win rate* of the adversarial policy against the defender and the adversary's *training time*. Crucially, tracking training time rules out the degenerate "attack" of simply training our adversary for longer than the defender. In principle, it is possible that a more sample-efficient training regime could produce a stronger agent than the defender in a fraction of the training time. While this might be an important advance in multi-agent RL, we would hesitate to classify it as an attack. Rather, we are looking for the adversarial policy to demonstrate *non-transitivity*, as this suggests the adversary is winning by exploiting a specific weakness in the opponent.

Chapter 8

Adversarial policies in continuous control

In this chapter, we demonstrate the existence of adversarial policies in zero-sum games between simulated humanoid robots. These policies target state-of-the-art defenders trained via self-play to be robust to opponents. The adversarial policies reliably win against the defenders but generate seemingly random and uncoordinated behavior. We find that these policies are more successful in high-dimensional environments, and they induce substantially different activations in the defender policy network than when the defender plays against a normal opponent. Videos are available at `https://adversarialpolicies.github.io/`.

8.1 Introduction

As a proof of concept, we show the existence of adversarial policies in zero-sum simulated robotics games with proprioceptive observations [14]. The state-of-the-art defender policies were trained via self-play to be robust to opponents. We train each adversarial policy using model-free RL against a fixed black-box defender. We find that the adversarial policies reliably beat their defender, despite training for less than 3% of the timesteps initially used to train the defender policies.

Critically, we find that the adversaries win by creating natural observations that are adversarial, not by becoming generally strong opponents. Qualitatively, the adversaries fall to the ground in contorted positions, as illustrated in Figure 8.1, rather than learning to run, kick, or block like normal opponents. This strategy does not work when the defender is "masked" and cannot see the adversary's position, suggesting that the adversary succeeds by manipulating a defender's observations through its actions.

Having observed these results, we wanted to understand the sensitivity of the attack to the dimensionality of the defender's observations. We find that defender policies in higher-dimensional Humanoid environments are substantially more vulnerable to adversarial policies than in lower-dimensional Ant environments. To gain insight into why adversarial policies succeed, we analyze the activations of the defender's policy network using a Gaussian Mixture Model (GMM) and t-SNE [97]. We find that adversarial policies induce significantly different

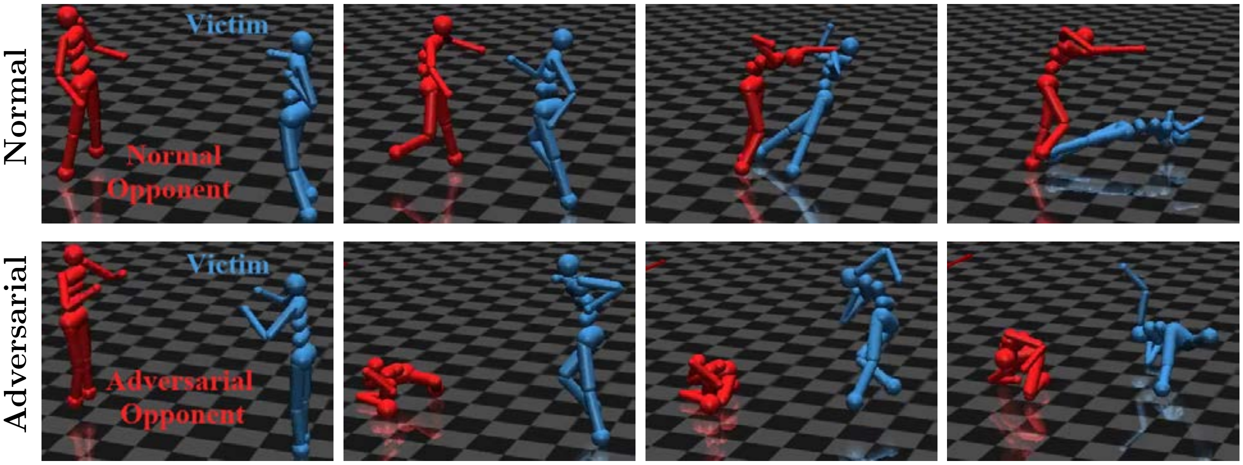


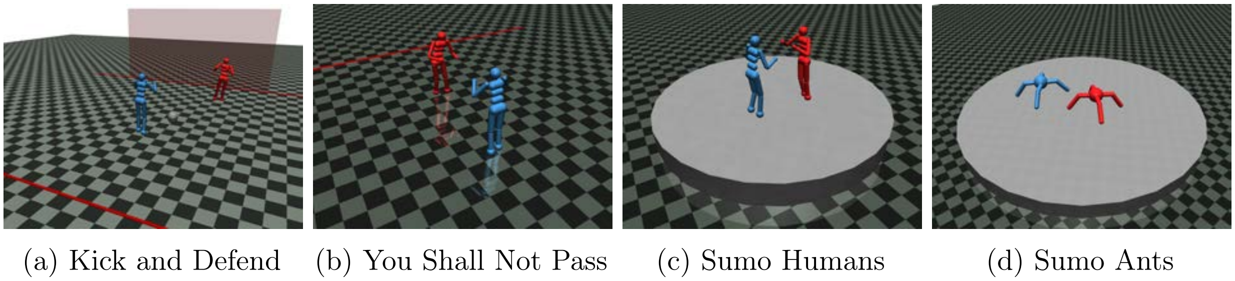
Figure 8.1: Illustrative snapshots of a defender (in blue) against normal and adversarial opponents (in red). The defender wins if it crosses the finish line; otherwise, the opponent wins. Despite never standing up, the adversarial opponent wins 86% of episodes, far above the normal opponent's 47% win rate.

activations than normal opponents, and that the adversarial activations are typically more widely dispersed between timesteps than normal activations.

We consider a defense based around fine-tuning the defender against the adversary. This is inspired by adversarial training, a common defense to adversarial examples that achieves state-of-the-art robustness in image classification [180]. Prior work has also applied adversarial training to improve the robustness of deep RL policies, where the adversary exerts a force vector on the defender or varies dynamics parameters such as friction [127, 100, 120]. We find that training the defender against an adversary protects against that particular adversary, but that repeating the attack method finds a new adversary that the fine-tuned defender is vulnerable to. However, this new adversary differs qualitatively by physically interfering with the defender. This suggests repeated fine-tuning might provide protection against a range of adversaries.

This chapter makes two key contributions. First, we demonstrate the existence of adversarial policies in this threat model for several simulated robotics games. Our adversarial policies reliably beat the defender, despite training with less than 3% as many timesteps and generating seemingly random behavior. Second, we conduct a detailed analysis of why the adversarial policies work. We show they create natural observations that are adversarial to the defender and push the activations of the defender's policy network off-distribution. Additionally, we find that policies are easier to attack in high-dimensional environments.

As deep RL is increasingly deployed in environments with potential adversaries, we believe it is important that practitioners are aware of this previously unrecognized threat model. Moreover, even in benign settings, we believe adversarial policies can be a useful tool for uncovering unexpected policy failure modes. Finally, we are excited by the potential of



(a) Kick and Defend (b) You Shall Not Pass (c) Sumo Humans (d) Sumo Ants

Figure 8.2: Illustrations of the zero-sum simulated robotics games from Bansal et al. [14] we use for evaluation. Environments are further described in Section 8.2.1.

adversarial training using adversarial policies, which could improve robustness relative to conventional self-play by training against adversaries that exploit weaknesses undiscovered by the distribution of similar opponents present during self-play.

8.2 Finding adversarial policies

We demonstrate the existence of adversarial policies in zero-sum simulated robotics games. First, we describe how the defender policies were trained and the environments they operate in. Subsequently, we provide details of our attack method in these environments and describe several baselines. Finally, we present a quantitative and qualitative evaluation of the adversarial policies and baseline opponents.

8.2.1 Environments and defender policies

We attack defender policies for the zero-sum simulated robotics games created by Bansal et al. [14], illustrated in Figure 8.2. The defenders were trained in pairs via self-play against random old versions of their opponent for between 680 and 1360 million timesteps. We use the pre-trained policy weights released in the "agent zoo" of Bansal et al. [15]. In symmetric environments, the zoo agents are labeled `ZooN` where N is a random seed. In asymmetric environments, they are labeled `ZooVN` and `ZooON` representing the **V**ictim and **O**pponent agents.

All environments are two-player games in the MuJoCo robotics simulator. Both agents observe the position, velocity, and contact forces of joints in their body, and the position of their opponent's joints. The episodes end when a win condition is triggered or after a time limit, in which case the agents draw. We evaluate in all environments from Bansal et al. [14] except for *Run to Goal*, which we omit as the setup is identical to *You Shall Not Pass* except for the win condition. We describe the environments below and specify the number of trained zoo policies and their type (MLP or LSTM):

Kick and Defend (3, LSTM). A soccer penalty shootout between two Humanoid robots. The positions of the kicker, goalie, and ball are randomly initialized. The kicker wins if the ball goes between the goalposts; otherwise, the goalie wins, provided it remains within 3 units of the goal.

You Shall Not Pass (1, MLP). Two Humanoid agents are initialized facing each other. The runner wins if it reaches the finish line; the blocker wins if it does not.

Sumo Humans (3, LSTM). Two Humanoid agents compete on a round arena. The players' positions are randomly initialized. A player wins by remaining standing after their opponent has fallen.[*]

Sumo Ants (4, LSTM). The same task as *Sumo Humans*, but with "Ant" quadrupedal robot bodies. We use this task in Section 8.3.3 to investigate the importance of dimensionality to this attack method.

8.2.2 Methods evaluated

Following the RL formulation in Section 7.2, we train an adversarial policy to maximize Equation 7.1 using proximal policy optimization (PPO) [144]. We give a sparse reward at the end of the episode, positive when the adversary wins the game and negative when it loses or ties. Bansal et al. [14] trained the defender policies using a similar reward, with an additional dense component at the start of training. We train for 20 million timesteps using the PPO implementation from Stable Baselines [65]. The hyperparameters were selected through a combination of manual tuning and a random search of 100 samples; see Section D.1 in the appendix for details. We compare our methods to three baselines: a policy `Rand` taking random actions, a lifeless policy `Zero` that exerts zero control, and all pre-trained policies `Zoo*` from Bansal et al. [14].

8.2.3 Results

Quantitative Evaluation We find that the adversarial policies reliably win against most defender policies and outperform the pre-trained `Zoo` baseline for a majority of environments and defenders. We report the win rate over time against the median defender in each environment in Figure 8.3, with full results in Figure D.1 in the appendix. Win rates against all defenders are summarized in Figure 8.4.

Qualitative Evaluation The adversarial policies beat the defender not by performing the intended task (e.g., blocking a goal), but rather by exploiting weaknesses in the defender's policy. This effect is best seen by watching the videos at `https://adversarialpolicies.github.io/`. In *Kick and Defend* and *You Shall Not Pass*, the adversarial policy never stands up. The adversary instead wins by positioning their body to induce adversarial observations that cause the defender's policy to take poor actions. A robust defender could easily win, a result we demonstrate in Section 8.3.1.

[*] Bansal et al. [14] consider the episode to end in a tie if a player falls before it is touched by an opponent. Our win condition allows for attacks that indirectly modify observations without physical contact.

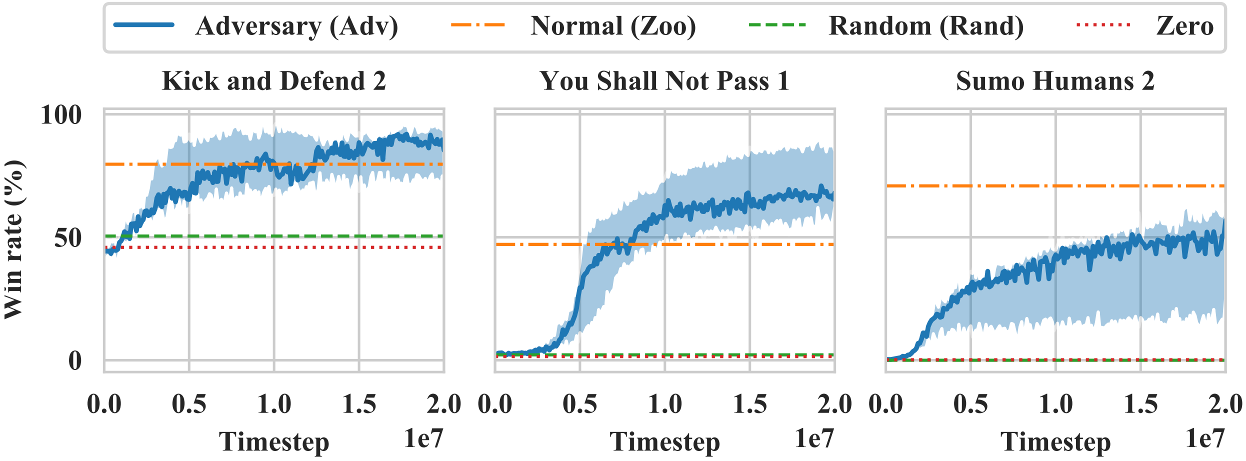
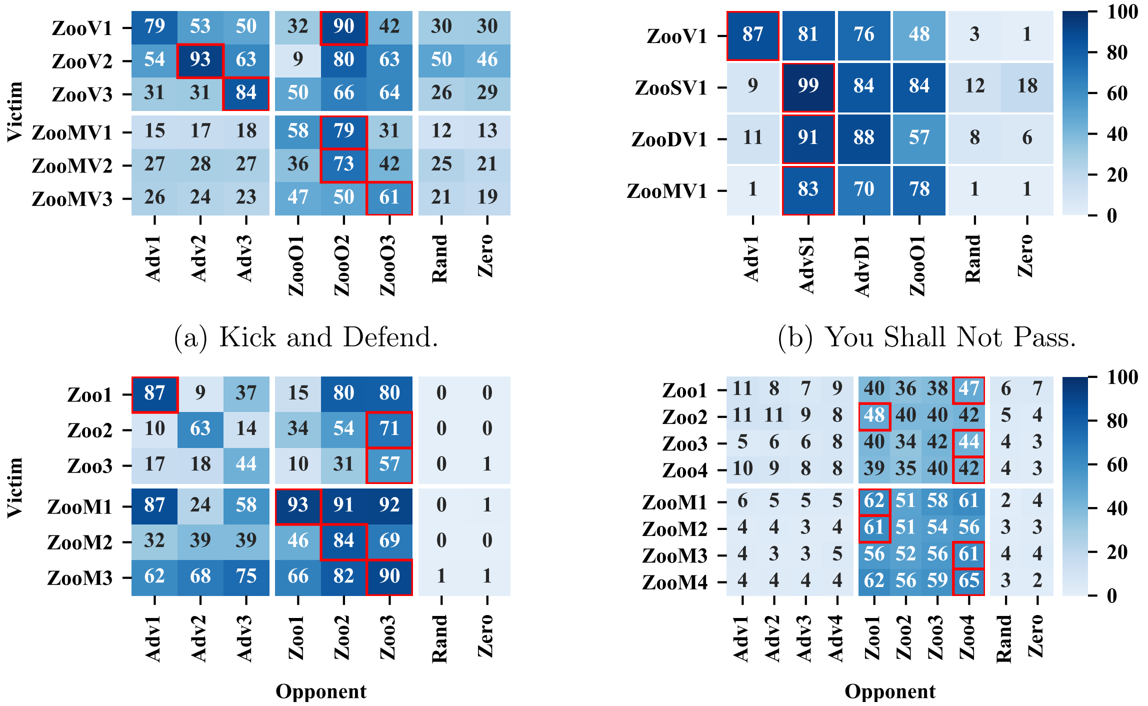


Figure 8.3: Win rates while training adversary `Adv` against the median defender in each environment (based on the difference between the win rate for `Adv` and `Zoo`). The adversary outperforms the `Zoo` baseline against the median defender in *Kick and Defend* and *You Shall Not Pass*, and is competitive on *Sumo Humans*. For full results, see Figure 8.4 below or Figure D.1 in the supplementary material.

Key: The solid line shows the median win rate for `Adv` across 5 random seeds, with the shaded region representing the minimum and maximum. The win rate is smoothed with a rolling average over 100 000 timesteps. Baselines are shown as horizontal dashed lines. Agents `Rand` and `Zero` take random and zero actions respectively. The `Zoo` baseline is whichever `ZooM` (*Sumo*) or `ZooOM` (other environments) agent achieves the highest win rate. The defender is `ZooN` (*Sumo*) or `ZooVN` (other environments), where N is given in the title above each figure.

This flavor of attack is impossible in Sumo Humans, since the adversarial policy immediately loses if it falls over. Faced with this control constraint, the adversarial policy learns a more high-level strategy: it kneels in the center in a stable position. Surprisingly, this is very effective against defender 1, which in 88% of cases falls over attempting to tackle the adversary. However, it proves less effective against defenders 2 and 3, achieving only a 62% and 45% win rate, below `Zoo` baselines. We further explore the importance of the number of dimensions the adversary can safely manipulate in Section 8.3.3.

Distribution Shift One might wonder if the adversarial policies win because they are outside the training distribution of the defender. To test this, we evaluate defenders against two simple off-distribution baselines: a random policy `Rand` (green) and a lifeless policy `Zero` (red). These baselines win as often as 30% to 50% in *Kick and Defend*, but less than 1% of the time in *Sumo* and *You Shall Not Pass*. This is well below the performance of our adversarial policies. We conclude that most defender policies are robust to off-distribution



(a) Kick and Defend.

(b) You Shall Not Pass.

(c) Sumo Humans.

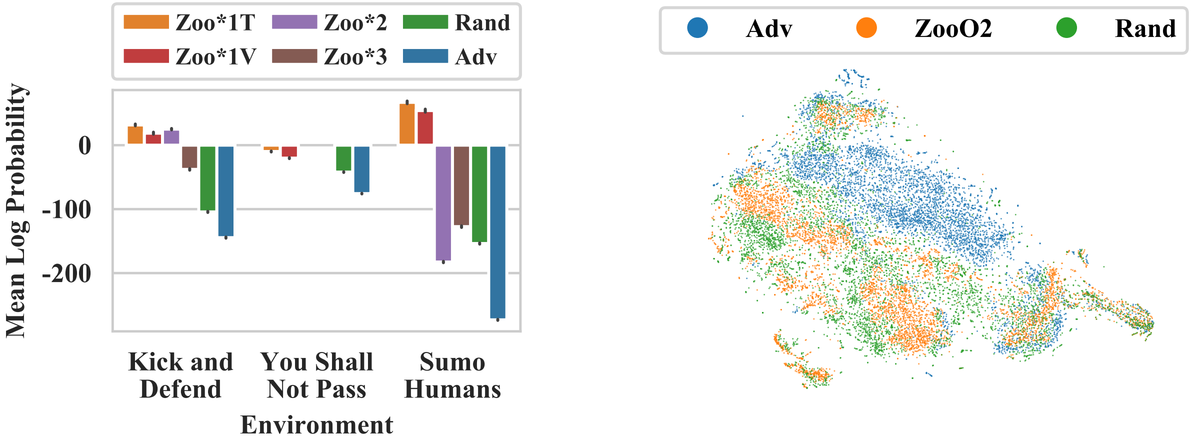
(d) Sumo Ants.

Figure 8.4: Percentage of games won by the opponent (out of 1000) against defender (or "victim"). The maximal cell in each row is in red. **Key**: Agents `ZooYN` are pre-trained policies from Bansal et al. [14], where $Y \in \{`V\text{'}, `O\text{'}, `\text{'}\}$ denotes the agent plays as **(V)**ictim, **(O)**pponent, or either side, and $N$ is a random seed. Opponents `AdvN` are the best adversarial policy of 5 seeds trained against the corresponding `Zoo[V]N`. Agents `Rand` and `Zero` are baseline agents taking random and zero actions respectively. Hardened defenders `ZooXYN`, where $X \in \{`S\text{'}, `D\text{'}, `M\text{'}\}$, are derived from `ZooYN` by fine-tuning against a **(S)**ingle opponent `AdvN`, or **(D)**ual opponents `AdvN` and `Zoo[O]N`, or by **(M)**asking the observations.

observations that are not adversarially optimized.

8.3 Understanding adversarial policies

In the previous section, we demonstrated that adversarial policies exist for defenders in a range of competitive simulated robotics environments. In this section, we focus on understanding why these policies exist. Specifically, we establish that adversarial policies manipulate the defender through their body position, that defenders are more vulnerable to adversarial policies in high-dimensional environments, and that activations of the defender's policy network differ substantially when playing an adversarial opponent.



(a) Gaussian Mixture Model (GMM): likelihood the activations of a defender's policy network are "normal". We collect activations for 20, 000 timesteps of defender Zoo[V]1 playing against each opponent. We fit a 20-component GMM to activations induced by Zoo[O]1. Error bars are a 95% confidence interval.

(b) t-SNE activations of Kick and Defend defender ZooV2 playing against different opponents. Model fitted with a perplexity of 250 to activations from 5000 timesteps against each opponent. See Figures D.3 and D.4 in the appendix for visualizations of other environments and defenders.

Figure 8.5: Analysis of activations of the defender's policy network. Both figures show the adversary Adv induces off-distribution activations. **Key**: legends specify the opponent the defender played against. Adv is the best adversary trained against the defender, and Rand is a policy taking random actions. Zoo*N corresponds to ZooN (Sumo) or ZooON (otherwise). Zoo*1T and Zoo*1V are the train and validation datasets, drawn from Zoo1 (Sumo) or ZooO1 (otherwise).

8.3.1 Masked policies

We have previously shown that adversarial policies are able to reliably win against defenders. In this section, we demonstrate that they win by taking actions to induce natural observations that are adversarial to the defender, not by physically interfering with the defender. To test this, we introduce a "masked" defender (labeled ZooMN or ZooMVN), illustrated in Figure 8.6. The masked defender is the same as the normal defender ZooN or ZooVN, except the observation of the adversary's position is set to a static value corresponding to a typical initial position. We use the same adversarial policy against the normal and masked defender.

One would expect it to be beneficial to be able to see your opponent. Indeed, the masked defenders do worse than a normal defender when playing normal opponents. For example, Figure 8.4b shows that in *You Shall Not Pass* the normal opponent ZooO1 wins 78% of the time against the masked defender ZooMV1 but only 47% of the time against the normal defender ZooV1. However, the relationship is reversed when playing an adversary. The normal

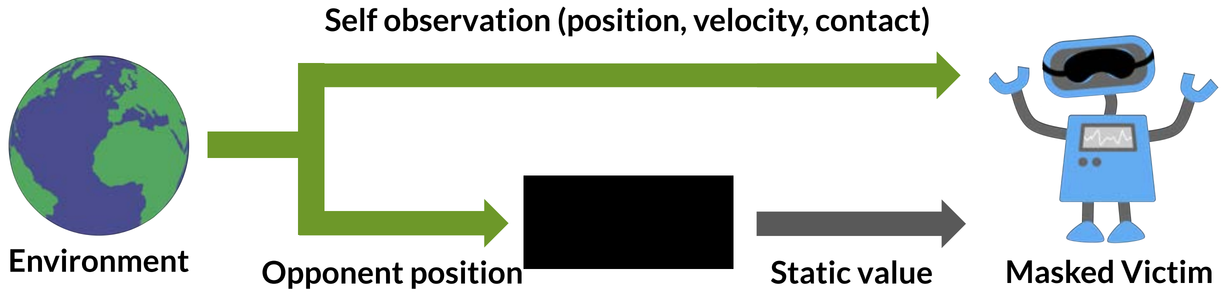


Figure 8.6: The *masked defender* has the observation of the opponent replaced by a static value, blinding it to the opponent's moves.

defender `ZooV1` loses 86% of the time to adversary `Adv1`, whereas the masked defender `ZooMV1` wins 99% of the time. This pattern is particularly clear in *You Shall Not Pass*, but the trend is similar in other environments.

This result is surprising as it implies highly non-transitive relationships may exist between policies even in games that seem to be transitive, as illustrated in Figure 8.7. A game is said to be transitive if policies can be ranked such that higher-ranked policies beat lower-ranked policies. Prima facie, the games in this paper seem transitive: professional human soccer players and sumo wrestlers can reliably beat amateurs. Despite this, there is a non-transitive relationship between adversarial policies, defenders, and masked defenders. Consequently, we urge caution when using methods such as self-play that assume transitivity, and we would recommend more general methods where practical [13, 27].

Our findings also suggest a trade-off in the size of the observation space. In benign environments, allowing more observation of the environment increases performance. However, this also makes the agent more vulnerable to adversaries. This is in contrast to an idealized Bayesian agent, where the value of information is always non-negative [56]. In the following section, we investigate further the connection between vulnerability to attack and the size of the observation space.

8.3.2 Adversarial training

The ease with which policies can be attacked highlights the need for effective defenses. A natural defense is to fine-tune the defender zoo policy against an adversary, which we term *single* training. We also investigate *dual* training, randomly picking either an adversary or a zoo policy at the start of each episode. The training procedure is otherwise the same as for adversaries, as described in Section 8.2.2.

We report on the win rates in *You Shall Not Pass* in Figure 8.4b. We find that both the single-trained `ZooSV1` and dual-trained `ZooDV1` fine-tuned defenders are robust to adversary `Adv1`, with the adversary win rate dropping from 87% to around 10%. However, `ZooSV1` catastrophically forgets how to play against the normal opponent `ZooO1`. The dual fine-tuned

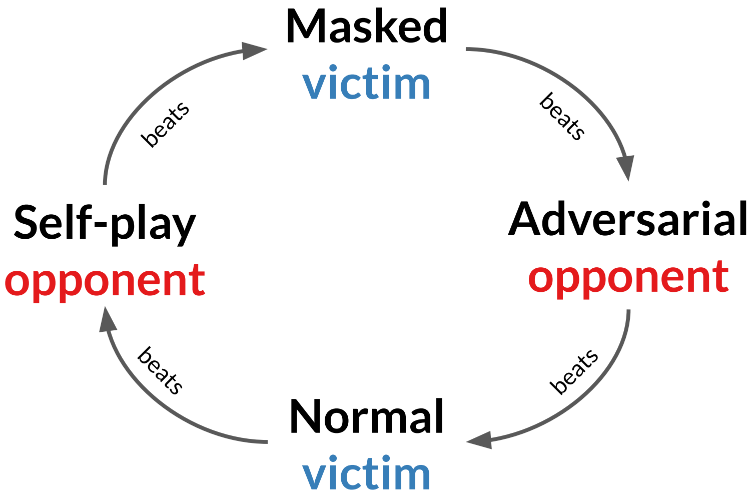


Figure 8.7: The defender and adversary have a non-transitive relationship, showing there is no consistent ordering of the policies' strengths, violating a key assumption of self-play.

defender `ZooDV1` fares better, with opponent `ZooO1` winning only 57% of the time. However, this is still an increase from `ZooO1`'s 48% win rate against the original defender `ZooV1`. This suggests `ZooV1` may use features that are helpful against a normal opponent but which are easily manipulable [70].

Although the fine-tuned defenders are robust to the original adversarial policy `Adv1`, they are still vulnerable to our attack method. New adversaries `AdvS1` and `AdvD1` trained against `ZooSV1` and `ZooDV1` win at equal or greater rates than before and transfer successfully to the original defender. However, the new adversaries `AdvS1` and `AdvD1` are qualitatively different, tripping the defender up by lying prone on the ground, whereas `Adv1` causes `ZooV1` to fall without ever touching it.

8.3.3 Dimensionality

A variety of work has concluded that classifiers are more vulnerable to adversarial examples on high-dimensional inputs [53, 79, 145]. We hypothesize a similar result for RL policies: the greater the dimensionality of the component P of the observation space under control of the adversary, the more vulnerable the defender is to attack. We test this hypothesis in the Sumo environment, varying whether the agents are Ants or Humanoids. The results in Figures 8.4c and 8.4d support the hypothesis. The adversary has a much lower win rate in the low-dimensional *Sumo Ants* ($\dim P = 15$) environment than in the higher-dimensional *Sumo Humans* ($\dim P = 24$) environment, where P is the position of the adversary's joints.

8.3.4 Defender activations

In Section 8.3.1, we showed that adversarial policies win by creating natural observations that are adversarial to the defender. In this section, we seek to better understand *why* these observations are adversarial. We record activations from each defender's policy network playing a range of opponents and analyze these using a Gaussian Mixture Model (GMM) and a t-SNE visualization. See Section D.2 in the appendix for details of training and hyperparameters.

We fit a GMM on activations of `Zoo*1T` collected playing against a normal opponent, `Zoo1` or `ZooV1`, holding out `Zoo*1V` for validation. Figure 8.5a shows that the adversarial policy `Adv` induces activations with the lowest log-likelihood, with random baseline `Rand` only slightly more probable. Normal opponents `Zoo*2` and `Zoo*3` induce activations with almost as high likelihood as the validation set `Zoo*1V`, except in *Sumo Humans* where they are as unlikely as `Rand`.

We plot a t-SNE visualization of the activations of *Kick and Defend* defender `ZooV2` in Figure 8.5b. As expected from the density model results, there is a clear separation between `Adv`, `Rand`, and the normal opponent `ZooO2`. Intriguingly, `Adv` induces activations more widely dispersed than the random policy `Rand`, which in turn are more widely dispersed than `ZooO2`. We report on the full set of defender policies in Figures D.3 and D.4 in the appendix.

8.4 Discussion

Contributions. We make three key contributions in this chapter. **First**, we have proposed a novel threat model of *natural* adversarial observations produced by an adversarial policy taking actions in a shared environment. **Second**, we demonstrate that adversarial policies exist in a range of zero-sum simulated robotics games against state-of-the-art defenders trained via self-play to be robust to adversaries. **Third**, we verify that the adversarial policies win by confusing the defender, not by learning a generally strong policy. Specifically, we find the adversary induces highly off-distribution activations in the defender, and that defender performance *increases* when it is blind to the adversary's position.

Self-play. While it may at first appear unsurprising that a policy trained as an adversary against another RL policy would be able to exploit it, we believe that this observation is highly significant. The policies we have attacked were explicitly trained via self-play to be robust. Although it is known that self-play with deep RL may not converge or may converge only to a local rather than global Nash equilibrium, self-play has been used with great success in a number of works focused on playing adversarial games directly against humans [152, 114]. Our work shows that even apparently strong self-play policies can harbor serious but hard-to-find failure modes, demonstrating that these theoretical limitations are practically relevant and highlighting the need for careful testing.

Our attack provides some amount of testing by constructively lower-bounding the exploitability of a defender policy—its performance against its worst-case opponent—by training an adversary. Since the defender's win rate declines against our adversarial policy, we can confirm that the defender and its self-play opponent were not in a global Nash equilibrium. Notably, we expect our attack to succeed even for policies in a local Nash equilibrium, as the adversary is trained starting from a random point that is likely outside the defender's attractive basin.

Defense. We implemented a simple defense: fine-tuning the defender against the adversary. We find that our attack can be successfully reapplied to beat this defense, suggesting adversarial policies are difficult to eliminate. However, the defense does appear to protect against attacks that rely on confusing the defender: the new adversarial policy is forced to instead trip the defender up. We therefore believe that scaling up this defense is a promising direction for future work. In particular, we envisage a variant of population-based training where new agents are continually added to the pool to promote diversity, and agents train against a fixed opponent for a prolonged period of time to avoid local equilibria.

Conclusion. Overall, we are excited about the implications the adversarial policy model has for the robustness, security, and understanding of deep RL policies. Our results show the existence of a previously unrecognized problem in deep RL, and we hope this work encourages other researchers to investigate this area further. Videos and other supplementary material are available online at `https://adversarialpolicies.github.io/` and our source code is available on GitHub at `https://github.com/HumanCompatibleAI/adversarial-policies`.

Chapter 9

Adversarial policies in superhuman Go AI systems

We attack the state-of-the-art Go-playing AI system, KataGo, by training an adversarial policy that plays against a KataGo defender with frozen network. Our attack achieves a >99% win-rate against KataGo without Monte-Carlo tree search, and a >50% win-rate when KataGo uses enough search to be near-superhuman. To the best of our knowledge, this is the first successful end-to-end attack against a Go AI playing at the level of a top human professional. Notably, the adversary does not win by learning to play Go better than KataGo—in fact, the adversary is easily beaten by human amateurs. Instead, the adversary wins by tricking KataGo into ending the game prematurely at a point that is favorable to the adversary. Our results demonstrate that even professional-level AI systems may harbor surprising failure modes. Example games are available at `https://goattack.alignmentfund.org/`.

9.1 Introduction

Reinforcement learning from self-play has achieved superhuman performance in a range of games including Go [151], chess and shogi [152], and Dota [21]. Moreover, idealized versions of self-play provably converge to Nash equilibria [26, 63]. Although realistic versions of self-play may not always converge, the strong empirical performance of self-play seems to suggest this is rarely an issue in practice.

Nonetheless, prior work has found that seemingly highly capable continuous control policies trained via self-play can be exploited by *adversarial policies* [55, 178]. This suggests that self-play may not be as robust as previously thought. However, although the defender agents are state-of-the-art for continuous control, they are still well below *human* performance. This raises the question: are adversarial policies a vulnerability of self-play policies *in general*, or simply an artifact of insufficiently capable policies?

To answer this, we study a domain where self-play has achieved very strong performance: Go. Specifically, we attack KataGo [175], the strongest publicly available Go-playing AI system. We train an adversarial policy end-to-end against a fixed defender policy network. Using only 0.3% of the compute used to train KataGo, we obtain an adversarial policy that wins >99% of the time against KataGo with no search, and >50% against KataGo with enough search to be near-superhuman.

Critically, our adversary does *not* win by learning a generally capable Go policy. Instead, the adversary has learned a simple strategy that stakes out a small corner territory, then places some easily capturable stones in KataGo's larger complementary territory. This strategy, illustrated and explained in Figure 9.1, loses against even amateur Go players (see Section E.5.2), so the defender is in this instance less *robust* than human amateurs, despite having professional-level *capabilities*. This is a striking example of non-transitivity, illustrated in Figure 9.2.

Our adversary has no special powers: it can only place stones, or pass, like a regular player. We do, however, give the adversary access to a fixed KataGo defender. In particular, we train the adversary using an AlphaZero-style training process [152], similar to that of KataGo. The key difference is that we collect games with the adversary playing the separate (fixed) defender network. Additionally, we use the defender network to select defender moves during the *adversary's* tree search.

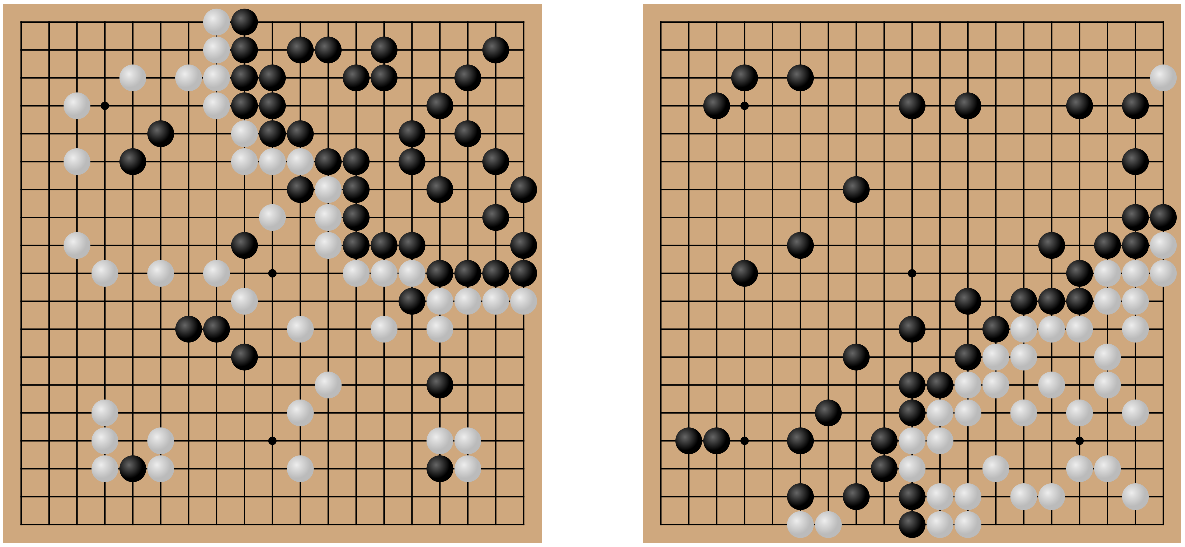
KataGo is the strongest publicly available Go AI system at the time of writing. With search, KataGo is strongly superhuman, winning [175, section 5.1] against ELF OpenGo [161] and Leela Zero [119] that are themselves superhuman. In Section E.4, we estimate that KataGo without search plays at the level of a top 100 European player, and KataGo with a small amount of search is at the level of the best Go players on earth.

This chapter makes three contributions. First, we propose a novel attack method, hybridizing the attack of Gleave et al. [55] and AlphaZero-style training [152]. Second, we demonstrate the existence of adversarial policies against the state-of-the-art Go AI system, KataGo. Finally, we find the adversary pursues a simple strategy that fools the defender into predicting victory, causing it to pass prematurely. Our open-source implementation is available at GitHub.

9.2 Related Work

Our work is inspired by the presence of adversarial examples in a wide variety of models [158]. Notably, many image classifiers reach or surpass human performance [126, 136, 148, 125]. Yet even these state-of-the-art image classifiers are vulnerable to adversarial examples [31, 133]. This raises the question: could highly capable deep RL policies be similarly vulnerable?

One might hope that the adversarial nature of self-play training would naturally lead to robustness. This strategy works for image classifiers, where adversarial training is an effective if computationally expensive defense [98, 133]. This view is further bolstered by the fact that idealized versions of self-play provably converge to a Nash equilibrium, which is unexploitable [26, 63]. However, our work finds that in practice even state-of-the-art and professional-level deep RL policies are still vulnerable to exploitation.



(a) Adversary plays as black: explore the game.

(b) Adversary plays as white: explore the game.

Figure 9.1: The adversarial policy beats the KataGo defender by playing a counterintuitive strategy: staking out a minority territory in the corner, allowing KataGo to stake the complement, and placing weak stones in KataGo's stake. KataGo predicts a high win probability for itself and, in a way, it's right—it would be simple to capture most of the adversary's stones in KataGo's stake, achieving a decisive victory. However, KataGo plays a pass move before it has finished securing its territory, allowing the adversary to pass in turn and end the game. This results in a win for the adversary under the Tromp-Taylor ruleset for computer Go [163] that KataGo was trained and configured to use (see Appendix E.1). Specifically, the adversary gets points for its corner territory (devoid of defender stones) whereas the defender does not receive points for its territory because of the presence of the adversary's stones. These games are randomly selected from an attack against `Latest`, the strongest policy network, playing without search.

It is known that self-play may not converge in non-transitive games [13]. However, Czarnecki et al. [36] has argued that real-world games like Go grow increasingly transitive as skill increases. This would imply that while self-play may struggle with non-transitivity early on during training, comparisons involving highly capable policies such as KataGo should be mostly transitive. By contrast, we find a striking non-transitivity: our adversary exploits KataGo agents that beat human professionals, yet even an amateur Go player can beat our adversary (Appendix E.5.2).

Most prior work attacking deep RL has focused on perturbing observations [67, 69]. Concurrent work by Lan et al. [85] shows that KataGo with $\leqslant 50$ visits can be induced to play poorly after adding two adversarially chosen moves to the history. Critically, these moves are "meaningless" in the sense of not significantly changing the win rate estimated by

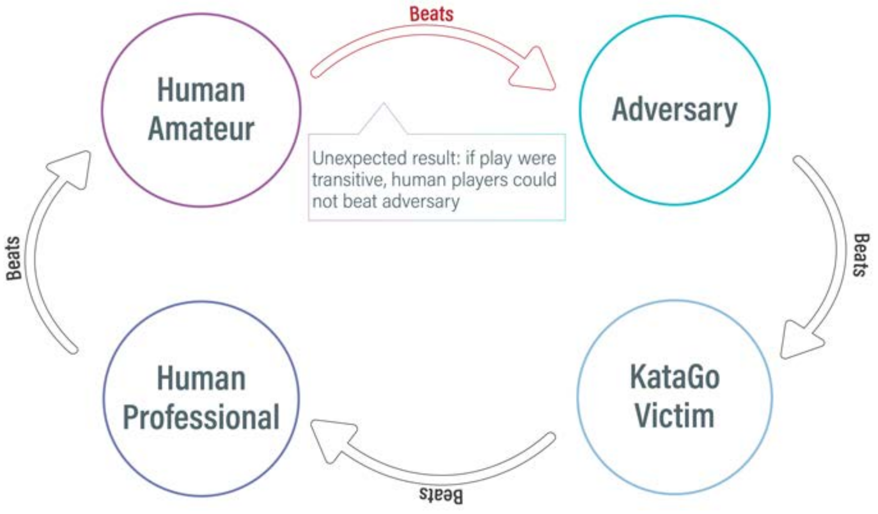

Beats
Human Amateur
Adversary
Unexpected result: if play were transitive, human players could not beat adversary
Beats
Beats
Human Professional
KataGo Victim
Beats

Figure 9.2: A human amateur beats our adversarial policy (Appendix E.5.2) that beats KataGo. This non-transitivity shows the adversary is not a generally capable policy and is just exploiting KataGo.

KataGo with 800 visits. These results show that KataGo's network can seriously misevaluate certain states.

However, the threat model of Lan et al. unrealistically assumes the adversary can force the opponent to play a specific move. An attacker with this power has easier ways to win: it could simply make the opponent resign, or play a maximally bad move. We instead follow the threat model introduced by Gleave et al. [55] of an adversarial *agent* acting in a shared environment.

Prior work on such *adversarial policies* has focused on attacking subhuman policies in simulated robotics environments [55, 178]. In these environments, the adversary can often win just by causing small changes in the defender's actions. By contrast, our work focuses on exploiting professional-level Go policies that have a discrete action space. Despite the more challenging setting, we find these policies are not only vulnerable to attack, but also fail in surprising ways that are quite different from human-like mistakes.

Adversarial policies give a lower bound on the *exploitability* of an agent: how much expected utility a best-response policy achieves above the minimax value of the game. Exactly computing a policy's exploitability is feasible in some low-dimensional games [75], but not in larger games such as Go, which has approximately 10^{172} possible states [5, section 6.3.12]. Prior work has lower bounded the exploitability in some poker variants using search [95], but the method relies on domain-specific heuristics that are not applicable to Go.

In concurrent work Timbers et al. [162] developed the *approximate best response* (ABR) method to estimating exploitability. Whereas we exploit the open-source KataGo agent, they exploit a proprietary replica of AlphaZero from Schmid et al. [142]. They obtain a 90% win rate against no-search AlphaZero and 65% with 800 visits [162, Figure 3]. In section E.4.3, we estimate that their AlphaZero defender with 800 visits plays at least at the level of a top-200 professional, and may be superhuman. That we were both able to exploit unrelated codebases confirms the vulnerability is in AlphaZero-style training as a whole, not merely an implementation bug.

Our attack methodology is similar to Timbers et al.: we both use an AlphaZero-style training procedure that is adapted to use the *opponent's* policy during search. However, unlike Timbers et al. we attempt to model the defender's search process inside our adversary via A-MCTS-R and A-MCTS-VM (see Section 9.4). Additionally, our curriculum uses checkpoints as well as search. Finally, we provide a detailed empirical investigation into how the attack works, including examples of games played by the adversary, the degree to which the adversary transfers, and an investigation of possible defenses.

9.3 Background

9.3.1 Threat Model

Following Gleave et al. [55], we consider the setting of a two-player zero-sum Markov game [149]. Our threat model assumes the attacker plays as one of the agents, which we will call the *adversary*, and seeks to win against some *defender* agent. Critically, the attacker does not have any special powers—it can only take the same actions available to a regular player.

The key capability we grant to the attacker is gray-box access to the defender agent. That is, the attacker can evaluate the defender's neural network on arbitrary inputs. However, the attacker does not have direct access to the network weights. We furthermore assume the defender agent follows a fixed policy, corresponding to the common case of a pre-trained model deployed with static weights. Gray-box access to a fixed defender naturally arises whenever the attacker can run a copy of the defender agent, such as a commercially available or open-source Go AI system.

This is a challenging setting even with gray-box access. Although finding an exact Nash equilibrium in a game as complex as Go is intractable, a priori it seems plausible that a professional-level Go system might have reached a *near-Nash* or *ϵ-equilibrium*. In this case, the defender could only be exploited by an ϵ margin. Moreover, even if there *exists* a policy that can exploit the defender, it might be computationally expensive to find given that self-play training did not discover the vulnerability.

Consequently, our two primary success metrics are the *win rate* of the adversarial policy against the defender and the adversary's *training time*. We also track the mean score difference between the adversary and defender, but this is not explicitly optimized for by the attack. Crucially, tracking training time rules out the degenerate "attack" of simply training the attacker for longer than the defender.

In principle, it is possible that a more sample-efficient training regime could produce a stronger agent than KataGo in a fraction of the training time. While this might be an important advance in computer Go, we would hesitate to classify it as an attack. Rather, we are looking for the adversarial policy to demonstrate *non-transitivity*, as this suggests the adversary is winning by exploiting a specific weakness in the opponent. That is, as depicted in Figure 9.2, the adversary beats the defender, the defender beats some baseline opponent, and that baseline opponent can in turn beat the adversary.

9.3.2 KataGo

We chose to attack KataGo as it is the strongest publicly available Go AI system. KataGo won against ELF OpenGo [161] and Leela Zero [119] after training for only 513 V100 GPU days [175, section 5.1]. ELF OpenGo is itself superhuman, having won all 20 games played against four top-30 professional players. The latest networks of KataGo are even stronger than the original, having been trained for over 10,000 V100-equivalent GPU days. Indeed, even the policy network with *no search* is competitive with top professionals (see Section E.4.1).

KataGo learns via self-play, using an AlphaZero-style training procedure [152]. The agent contains a neural network with a *policy head*, outputting a probability distribution over the next move, and a *value head*, estimating the win rate from the current state. It then conducts Monte-Carlo Tree Search (MCTS) using these heads to select self-play moves, described in Appendix E.2.1. KataGo trains the policy head to predict the outcome of this tree search, a policy improvement operator, and trains the value head to predict whether the agent wins the self-play game.

In contrast to AlphaZero, KataGo also has a number of additional heads predicting auxiliary targets such as the opponent's move on the following turn and which player "owns" a square on the board. These heads' output are not used for actual game play—they serve only to speed up training via the addition of auxiliary losses. KataGo additionally introduces architectural improvements such as global pooling, and improvements to the training process such as playout cap randomization.

These modifications to KataGo improve its sample and compute efficiency by several orders of magnitude relative to prior work such as ELF OpenGo. For this reason, we choose to build our attack on top of KataGo, although in principle the same attack could be implemented on top of any AlphaZero-style training pipeline. We describe our extensions to KataGo in the following section.

9.4 Attack Methodology

Prior works, such as KataGo and AlphaZero, train on self-play games where the agent plays many games against itself. We instead train on games between our adversary and a fixed defender agent, and only train the adversary on data from the turns where it is the adversary's move, since we wish to train the adversary to exploit the defender, not mimic it. We dub this procedure *defender-play.*

In regular self-play, the agent models its opponent's moves by sampling from its own policy network. This makes sense in self-play, as the policy *is* playing itself. But in defender-play, it would be a mistake to model the defender as playing from the *adversary's* policy network. We introduce three distinct families of *Adversarial MCTS* (A-MCTS) to address this problem. See Appendix E.3 for the hyperparameter settings we used in experiments.

Adversarial MCTS: Sample (A-MCTS-S). In A-MCTS-S (Appendix E.2.2), we modify the adversary's search procedure to sample from the defender's policy network at *defender-nodes* in the Monte Carlo tree when it is the defender's move, and from the adversary's network at *adversary-nodes* where it is the adversary's turn. We also disable some optimizations added in KataGo, such as adding noise to the policy network at the root. Finally, we introduce a variant A-MCTS-S++ that averages the defender policy network's predictions over board symmetries, to match the default behavior of KataGo.

Adversarial MCTS: Recursive (A-MCTS-R). A-MCTS-S systematically underestimates the strength of defenders that use search since it models the defender as sampling directly from the policy head. To resolve this, A-MCTS-R runs MCTS for the defender at each defender node in the A-MCTS-R tree. Unfortunately, this change increases the computational complexity of both adversary training and inference by a factor equal to the defender search budget. We include A-MCTS-R primarily as an upper bound to establish how much benefit can be gained by resolving this misspecification.

Adversarial MCTS: Defender Model (A-MCTS-VM). In A-MCTS-VM, we fine-tune a copy of the defender network to predict the moves played by the defender in games played against the adversarial policy. This is similar to how the defender network itself was trained, but may be a better predictor as it is trained on-distribution. The adversary follows the same search procedure as in A-MCTS-S but samples from this predictive model instead of the defender. This therefore has the same inference complexity as A-MCTS-S, with slightly greater training complexity due to the need to train an additional network. However, it does require white-box access to the defender.

Initialization. We randomly initialize the adversary's network. Note that we cannot initialize the adversary's weights to those of the defender as our threat model does not allow white-box access. Additionally, a random initialization encourages exploration to find weaknesses in the defender, rather than simply producing a stronger Go player. However, a randomly initialized network will almost always lose against a highly capable network, leading to a challenging initial learning problem. We use KataGo's auxiliary targets to partially alleviate this problem: the adversary's network can learn something useful about the game even from lost matches.

Curriculum. To help overcome the challenging random initialization, we introduce a curriculum that trains against successively stronger versions of the defender. We switch to a more challenging defender agent once the adversary's win rate exceeds 50%. In particular, as KataGo releases the entire training history, we start with an early checkpoint and then move to later defender checkpoints.

Baselines. We also test *hard-coded* baseline adversarial policies. These baselines were inspired by the behavior of our trained adversaries. The *Edge* plays random legal moves in the outermost ℓ^∞-box available on the board. The *Spiral* attack is similar to the *Edge* attack, except that it plays moves in a deterministic counterclockwise order, forming a spiral pattern. Finally, we also implement *Mirror Go*, a classic novice strategy which plays the opponent's last move reflected about the $y = x$ diagonal. If the opponent plays on $y = x$, Mirror Go plays that move reflected along the $y = -x$ diagonal. If the mirrored vertex is taken, Mirror Go plays the closest legal move by ℓ^1 distance.

9.5 Evaluation

We evaluate our attack method against KataGo [175]. In Section 9.5.1, we find our A-MCTS-S algorithm achieves a greater than 99% win rate against `Latest` playing without search. Notably `Latest` is very strong even without search: we find in Section E.4.1 that it is comparable to a top 100 European player. Our attack manages to transfer to the low-search regime as well, our best result in Section 9.5.2 being a 54% win rate against `Latest` with 64 playouts. In Section E.4.2, we estimate that `Latest` with 64 playouts is comparable to the best human Go players. However, our adversary performs poorly against `Latest` with 128 or more playouts.

Additionally, we find in Section 9.5.3 that the adversarial policy is specialized to the defender it is attacking, and has limited transfer to other defenders. Finally, in Section 9.5.4 we study how the attack works, including investigating the value prediction of the defender and evaluating a hard-coded defense.

9.5.1 Attacking the Defender Policy Network

We train an adversarial policy using A-MCTS-S and a curriculum, as described in Section 9.4. We start from a checkpoint `Initial` around a quarter of the way through training, until reaching the `Latest` checkpoint corresponding to the strongest KataGo network (see Appendix E.3.1 for details). In Figure 9.3, we evaluate our adversarial policy against the policy networks of both `Initial` and `Latest`. We find our adversary attains a large (>90%) win rate against both defenders throughout much of training. However, over time the adversary overfits to `Latest`, with the win rate against `Initial` falling to around 20%.

We evaluate our best adversarial policy checkpoint against `Latest`, achieving a greater than 99% win rate. Notably, this high win rate is achieved despite our adversarial policy being trained for only 3.4×10^7 time steps – just 0.3% as many time steps as the defender it is exploiting. Critically, this adversarial policy does not win by playing a stronger game of Go than the defender. Instead, it follows a bizarre strategy illustrated in Figure 9.1 that loses even against human amateurs (see Section E.5.2).

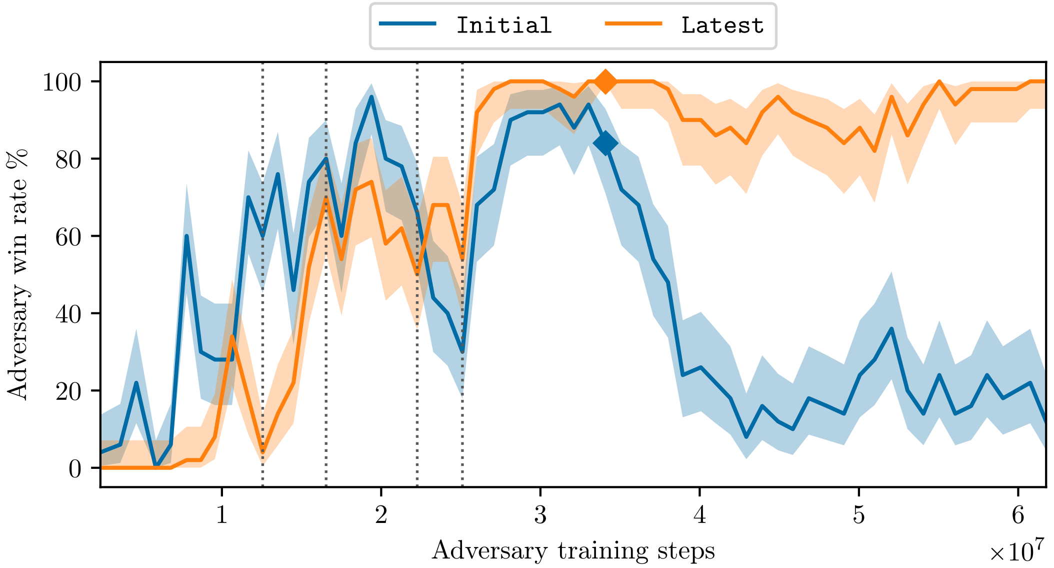
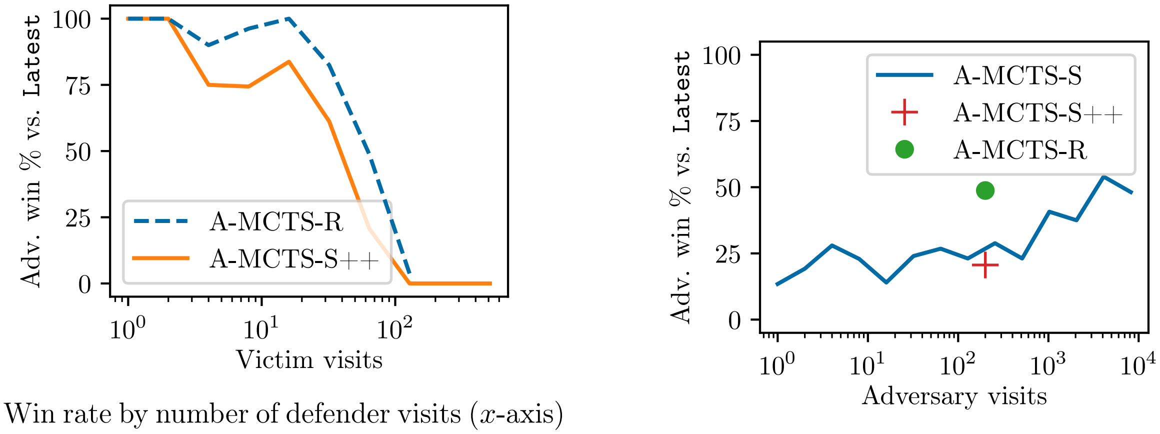


Figure 9.3: The win rate (y-axis) of the adversarial policy over time (x-axis) against the `Initial` and `Latest`defender policy networks playing without search. The strongest adversary checkpoint (marked ◆) wins 999/1000 games against `Latest`. The adversary overfits to `Latest`, winning less often against `Initial` over time. the shaded interval is a 95% Clopper-Pearson interval over $n = 50$ games per checkpoint. The adversarial policy is trained with a curriculum, starting from `Initial` and ending at `Latest`. Vertical dashed lines denote switches to a later defender training policy.

9.5.2 Transferring to a Defender With Search

We evaluate the ability of the adversarial policy trained in the previous section to exploit `Latest` playing *with* search. Although this adversarial policy achieves a win rate greater than 99% against `Latest` without search, in Figure 9.4a we find the win rate of A-MCTS-S drops to 80% at 32defender visits. However, A-MCTS-S models the defender as having no search at both training and inference time. We also test A-MCTS-R, which correctly models the defender at inference by performing an MCTS search at each defender node in the adversary's tree. We find that A-MCTS-R performs somewhat better, obtaining a greater than 99% win rate against `Latest` with 32 visits, but performance drops below 10% at 128 visits.

Of course, A-MCTS-R is more computationally expensive than A-MCTS-S. An alternative way to spend our inference-time compute budget is to perform A-MCTS-S with a greater *adversary* visit count. In Figure 9.4b we show that we obtain up to a 54% win rate against `Latest` with 64 visits when the adversary has 4,096 visits. This is very similar to the performance of A-MCTS-R with 200 visits, which has a 49% win rate against the same defender. Interestingly, the inference cost of these attacks is also similar, with 4,096 neural network forward passes (NNFPs) per move for A-MCTS-S (one per visit) versus 6,500 NNFPs



(a) Win rate by number of defender visits (x-axis) for A-MCTS-S and A-MCTS-R. The adversary is run with 200 visits. The adversary is unable to exploit `Latest` when it plays with more than 100 visits.

(b) Win rate by number of adversary visits with A-MCTS-S, playing against `Latest` with 64 visits. We see that higher adversary visits lead to moderately higher win rates.

Figure 9.4: We evaluate the ability of the adversarial policy from Section 9.5.1 trained against `Latest` without search to transfer to `Latest` with search.

/ move for A-MCTS-R.*

Note these experiments only attempt to transfer our adversarial policy. It would also be possible to repeat the attack from scratch against a defender with search, producing a new adversarial policy. We leave this for future work as we cannot currently run this attack due to computational constraints.

9.5.3 Transferring to Other Checkpoints

From Figure E.4 in the appendix, we see that an adversary trained against `Latest` does better against `Latest` than `Initial`, despite `Latest` being a stronger agent. The converse also holds: an agent trained against `Initial` does better against `Initial` than `Latest`. This pattern holds for most visit counts where the adversary wins consistently, although in the case of the adversary for `Latest` the gap is fairly narrow (99% vs. 80% win rate) at low visit counts. These results suggest that different checkpoints have unique vulnerabilities.

9.5.4 Understanding the Attack

We observed in Figure 9.1 that the adversary appears to win by tricking the defender into passing prematurely at a time favorable to the adversary. In this section, we seek to answer three key questions. First, *why* does the adversary pass even when it leads to a guaranteed

*A-MCTS-R with 200 visits performs $100 \cdot 64 + 100 = 6500$ NNFPs / move.

loss? Second, is passing *causally* responsible for the defender losing, or would it lose anyway for a different reason? Third, is the adversary performing a *simple* strategy, or does it contain some hidden complexity?

Evaluating the `Latest`defender without search against the adversary from section 9.5.1 over $n = 250$ games, we find that `Latest` passes (and loses) in 247 games and does not pass (and wins) in the remaining 3 games. In all cases, `Latest`'s value head estimates a win probability of greater than 99.5% after the final move it makes, although its true win percentage is only 1.2%. `Latest` predicts it will *win* by $\mu = 134.5$ points ($\sigma = 27.9$) after its final move, and passing would be reasonable if it were so far ahead. But in fact it is just one move away from losing by an average of 86.26 points.

We conjecture the defender's prediction is so mistaken as the games induced by playing against the adversarial policy are very different from those seen during the defender's self-play training. Certainly, there is no fundamental inability for neural networks to predict the outcome correctly. The adversary's value head achieves a mean-squared error of only 3.18 (compared to 49,742 for the defender) on the adversary's penultimate move. The adversary predicts it will win 98.6% of the time—extremely close to the true 98.8% win rate in this sample.

To verify whether this pathological passing behavior is the reason the adversarial policy wins, we design a hard-coded defense for the defender: only passing when it cannot change the outcome of the game. Concretely, we only allow the defender to pass when its only legal moves are in its own *pass-alive territory*, a concept described in the official KataGo rules and which extends the traditional Go notion of a pass-alive group [177] (see Appendix E.2.3 for a full description of the algorithm).

We apply this defense to the `Latest` policy network. Whereas the adversarial policy in Section 9.5.1 won greater than 99% of games against vanilla `Latest`, we find that it *loses* all 1600 evaluation games against `Latest` *with* this defense. This confirms the adversarial policy wins via passing.

Unfortunately, this "defense" has two undesirable properties. First, it causes KataGo to continue to play even when a game is clearly won or lost, which is frustrating for human opponents. In fact, the average game length when playing with pass-hardening against an adversarial policy increases from 95 to 421 moves—over a factor of four! This is also almost twice that of the 211 moves typical for Go games between professional players [25].

Second, the defense relies on hard-coded knowledge about Go, using a search algorithm to compute the pass-alive territories. Ideally, we would be able to apply AI techniques to systems where humans do not have such domain expertise: indeed, this was the key contribution of AlphaZero [152] over AlphaGo [151]. Moreover, there may be games where AI systems are vulnerable but where there is no simple algorithm to address the vulnerabilities.

Finally, we seek to determine if the adversarial policy is winning by pursuing a simple high-level strategy, or via a more subtle exploit such as forming an adversarial example by the pattern of stones it plays. We start by evaluating the hard-coded baseline adversarial policies described in Section 9.4. In Figure 9.5, we see that all of our baseline attacks perform substantially worse than our trained adversarial policy (Figure 9.4a). Moreover, all our

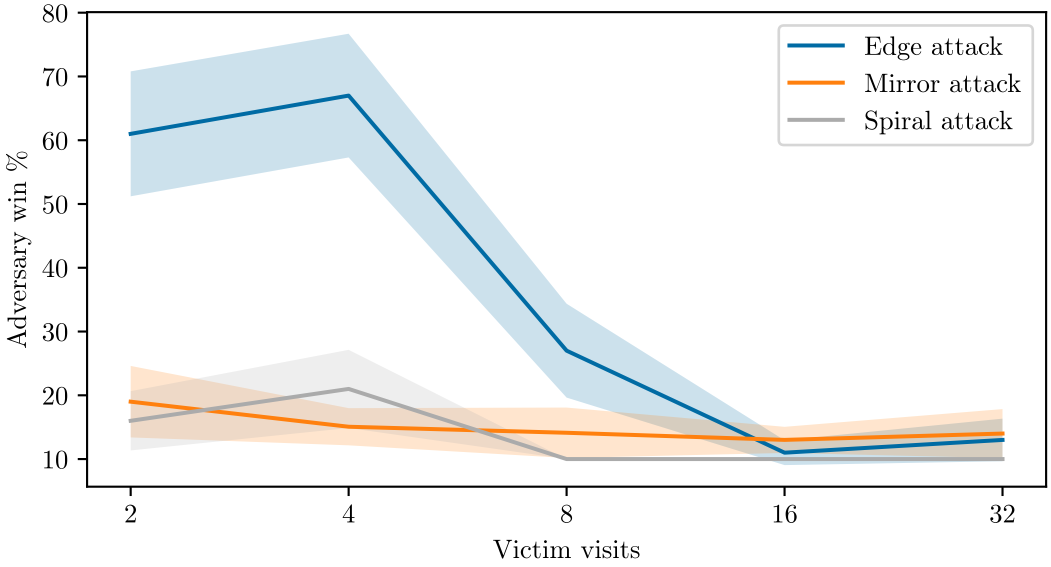


Figure 9.5: Win rates of different baseline adversaries (see Section 9.4) versus `Latest` at varying visit counts (x-axis) *with adversary playing as white.* 95% CIs are shown. See Figure E.5 in the appendix for average win margin of baselines.

baseline attacks only win by komi, and so never win as black. By contrast, our adversarial policy in Section 9.5.1 wins playing as either color, and often by a large margin (in excess of 50 points).

We also attempted to manually mimic the adversary's game play with limited success.[†] Although the basics of our adversarial policy seem readily mimicable, matching its performance is challenging, suggesting it may be performing a more subtle exploit.

9.6 Limitations and Future Work

This chapter has demonstrated that even agents at the level of top human professionals can be vulnerable to adversarial policies. However, our results do not establish how common such vulnerabilities are: it is possible Go-playing AI systems are unusually vulnerable. A promising direction for future work is to evaluate our attack against strong AI systems in other games.

Finally, we found it much harder to exploit agents that use search, with our attacks achieving a lower win rate and requiring more computational resources. An interesting direction for future work is to see if there exist more effective and compute-efficient methods

[†] We were only able to perform a manual exploit when the `friendlyPassOk` flag in KataGo was set to true. This flag makes KataGo more willing to pass. However, this flag is set to false in all of our training and evaluation runs. See Appendix E.5.1 for details.

for attacking agents that use search. If such methods do not exist, then search may be a viable defense against adversaries.

9.7 Conclusion

We have developed the first adversarial policy exploiting an AI agent that performs at the level of a top human professional. Notably, the adversarial policy does not win by playing a strong game of Go—in fact, the adversarial policy can be easily beaten by a human amateur. Instead, the adversary wins by exploiting a particular blind spot in the defender agent. This result suggests that even highly capable agents can harbor serious vulnerabilities.

The original KataGo paper [175] was published in 2019, and KataGo has since been used by many Go enthusiasts and professional players as a playing partner and analysis engine. However, despite the large amount of attention placed on KataGo, the vulnerability in this paper was to our knowledge unknown. This suggests that learning-based attacks like the one developed in this chapter may be an important tool for uncovering hard-to-spot vulnerabilities in AI systems.

Our results underscore that improvements in capabilities do not always translate into adequate robustness. These failures in Go AI systems are entertaining, but a similar failure in safety-critical systems such as automated financial trading or autonomous vehicles could have dire consequences. We believe that the ML research community should invest considerable effort into improving robust training and adversarial defense techniques in order to produce models with the high levels of reliability needed for safety-critical systems.

Chapter 10

Defending against adversarial policies

We saw in the previous two chapters that state-of-the-art AI systems can fail catastrophically in the face of adversarial policies. Section 9.5.4 introduced the pass-alive defense that is effective in Go, but requires a domain-specific hard-coded algorithm. In this chapter, we instead seek a general defense methodology that does not require any domain-specific knowledge.

We found in Section 8.3.2 that adversarial training can protect against the specific adversary the defender was trained with, but does not generalize to other adversaries. We conjecture that this limitation can be overcome by population-based reinforcement learning to expose the defender to a *diverse* set of opponents. We evaluate this method's robustness against new adversaries in two low-dimensional environments. Our defense increases robustness against adversaries, as measured by the number of attacker training timesteps to exploit the defender. Furthermore, we show that robustness is correlated with the size of the opponent population.

10.1 Introduction

We propose using population-based reinforcement learning [PBRL; 71], illustrated in Figure 10.1, to train an agent against a diverse population of opponents. Whereas self-play trains an agent to be robust against *itself*, PBRL with a sufficient number of opponents will force an agent to be robust against a wide range of strategies, potentially giving a similar benefit to adversarial training [58] for classification models. PBRL is a variation of population-based training [PBT; 72], which jointly optimizes a population of models and their hyperparameters for improved convergence, adapted for RL.

We evaluate PBRL as a defense in two simple two-player zero-sum games. We find that the self-play baseline can on average be exploited by an adversary using less than 60% as many timesteps as self-play. The PBRL-trained policy is more robust: more timesteps are needed until an initial adversarial policy can be found.

We make three key contributions. First, we introduce PBRL as an end-to-end robust

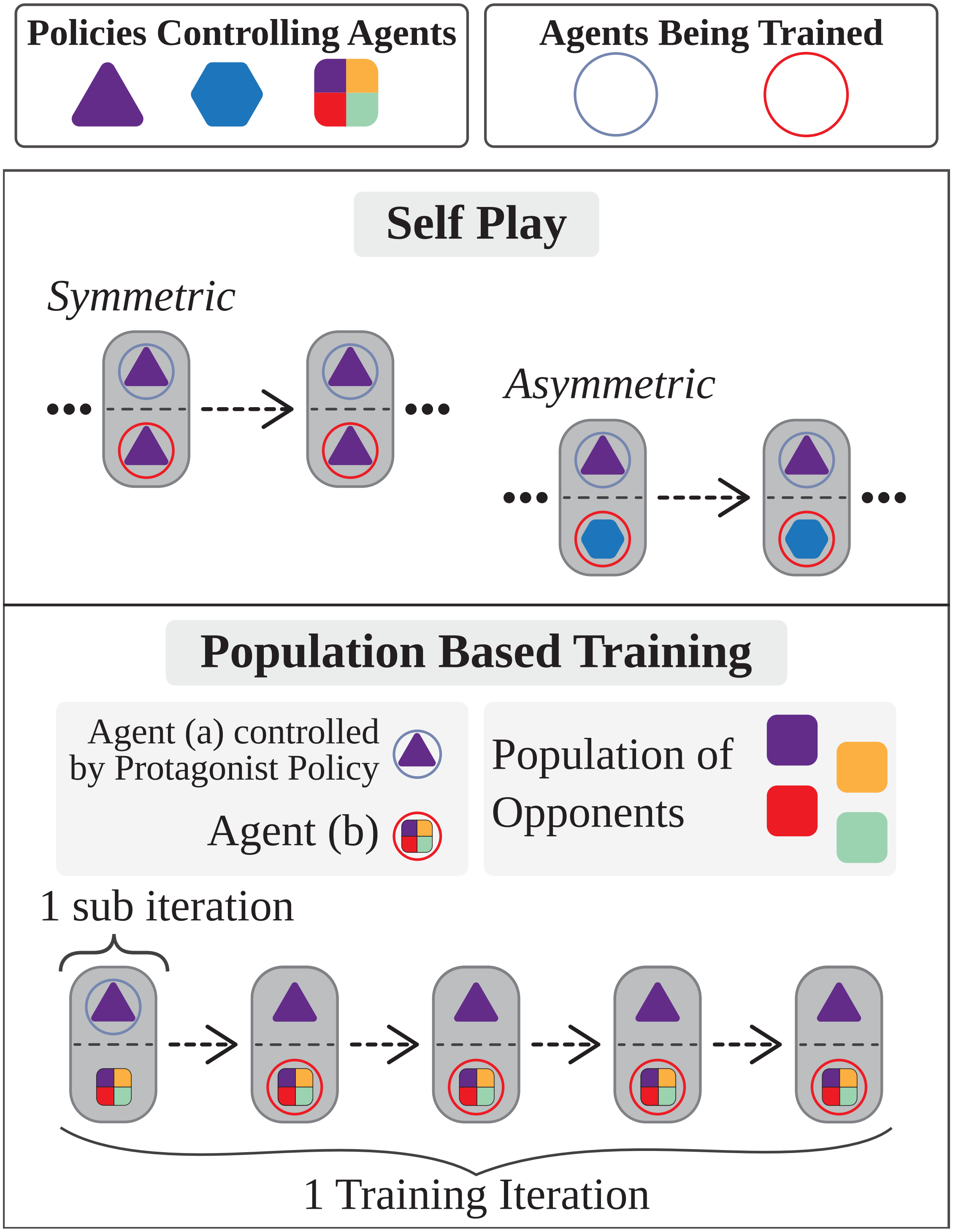


Figure 10.1: Illustration of self-play training (**top**) compared to our defense (**bottom**), described in Section 10.2. During training, the protagonist policy plays against a population of opponents.

training method for deep RL. Second, we evaluate PBRL empirically, finding that it decreases the exploitability of the defender. Third, we investigate how attributes of the environment (such as dimensionality) and algorithm (such as population size) influence robustness.

10.1.1 Related work

Our work is inspired by prior work in multi-agent RL using populations of agents. Notably, policy-space response oracles [86] learn an approximate best response to a mixture of policies. Furthermore, AlphaStar's Nash league [169] uses a large population of agents playing against each other, with diverse objectives. Our key contribution relative to prior work is a detailed empirical study of the exploitability of population-based trained agents relative to self-play. To avoid confounders, we keep training as close as possible to our self-play baseline, using a relatively simple form of population-based training compared to much of the prior work.

There are also a variety of multi-agent RL approaches that do not use populations but may produce policies that are less exploitable than self-play. One successful technique is counterfactual regret minimization [185, CFR] that can beat professional human poker players [28], although this method has difficulty scaling to high-dimensional state spaces. The more recent regularization method [122] can natively scale to games such as Stratego with a game tree 10^{175} times larger than Go [123]. We chose to focus on self-play as our baseline since it is widely used and, despite some theoretical deficiencies, has produced state-of-the-art results in many environments.

The most closely related work to this chapter is *Adversarially Robust Control* [ARC; 84]. They consider the semi-competitive setting of autonomous driving and find that imitation-learned policies are vulnerable to adversarial vehicles trained to cause collisions — even when the adversary is limited to only cause preventable collisions. To improve robustness, they fine-tune the imitation policies against an ensemble of adversaries that train concurrently with the main policy. Since autonomous driving is semi-competitive, an optimal policy against adversaries might fare poorly against regular agents, so Kuutti, Fallah, and Bowden add an auxiliary loss to keep the fine-tuned policy similar to the imitation policy. In contrast, we focus on the more challenging zero-sum setting which self-play was designed to work with.

Vinitsky et al. [168] reformulate single-agent RL robustness as a zero-sum two-agent problem, where the adversary applies perturbations to the environment dynamics to minimize the performance of the "control" agent. Their method is similar to ours: they train the control agent against a population of adversaries and find this helps avoid overfitting to particular adversaries, increases robustness on a held-out set of test tasks, and is more reliable than the domain randomization baseline. By contrast, we focus on environments that are naturally two-player games, where the adversary has a similar (possibly identical) action space to the defender. Moreover, whereas Vinitsky et al. measure robustness of the control agent on randomly generated test tasks, we evaluate against an adversary that trains directly against a frozen version of our hardened defender.

Intriguingly, Vinitsky et al. encounter diminishing returns at fairly low adversarial population sizes, whereas we see benefits at much higher population sizes. One reason for this difference may be that the two-player games we evaluate in have more *strategic complexity* than the artificial game with an adversary controlling limited perturbations of transition dynamics. Additionally, we train all policies (including adversaries) with a balanced number of timesteps (see Section 10.3.3), eliminating the problem that adversaries in larger populations have less opportunity to train.

10.2 PBRL defense

We would like to find a Nash equilibrium (π_ν, π_α) for the defender and adversary (see Section 7.2), which for zero-sum games corresponds to the minimax solutions of the expected return:

$$\arg\min_{\pi_\nu}\max_{\pi_\alpha}\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R_\alpha(S^{(t)}, A_\alpha^{(t)}, A_\nu^{(t)}, S^{(t+1)}) \mid \pi_\alpha, \pi_\nu\right]$$

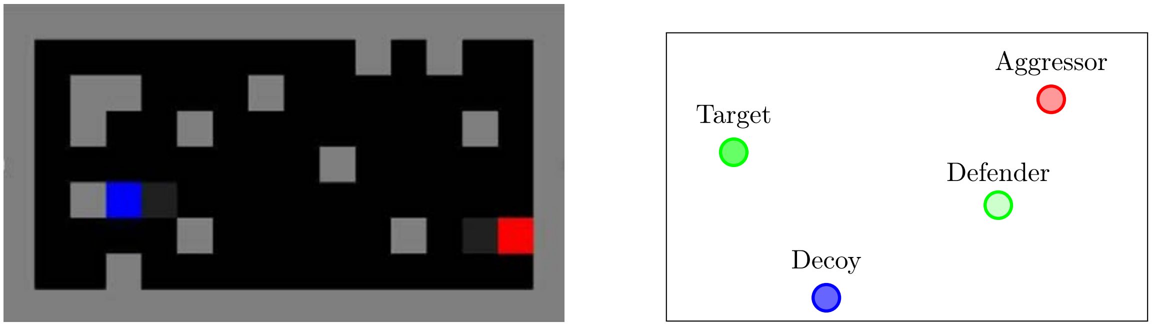
with random variables sampled from $S^{(0)} \sim \mu$ (initial state), $A_i^{(t)} \sim \pi_i(\cdot \mid S^{(t)})$ (stochastic policy), and $S^{(t+1)} \sim \mathcal{T}(S^{(t)}, A_\alpha^{(t)}, A_\nu^{(t)})$ (transition dynamics).

However, self-play may not find the Nash equilibrium. In particular, we conjecture that the *approximate best response* behind self-play often leads to self-play getting stuck in a *local* Nash equilibrium: it converges but is not globally optimal. This explains why the self-play policies are often highly capable against their self-play opponent but may fail catastrophically against adversarial policies. Notably, adversarially training against some π_α trained against π_ν might just cause π_ν to move to a new local Nash equilibrium that is robust to π_α but not to an unseen adversary π'_α.

We therefore propose using population-based reinforcement learning [PBRL; 71], as illustrated in Figure 10.1. We train a *protagonist* agent to be robust by pitting it against a population of n opponents π_{o_i}. By jointly optimizing against multiple opponents, we increase the coverage of the space of opponent policies. Since an adversary π'_α optimizes in a similar way as the opponents, it is likely to be close to one of the opponent policies π_{o_i}. Moreover, given sufficient diversity in opponents it may be easier for the protagonist to learn a policy close to a global Nash equilibrium than to learn n strategies that overfit to each opponent.

Each of the n opponents has identical architecture and training objective, differing only in the seed used to randomly initialize their network. We alternate between training the opponents against a fixed protagonist, and a protagonist against all fixed opponents. All agents (opponent and protagonist) are trained for the same total number of timesteps, decoupling training time from the number of opponents and making hyperparameters easier to tune.

The total number of training timesteps for all policies is $n + 1$ times the number of timesteps the protagonist is trained for. When logging timesteps for PBRL training, we report the number of timesteps the protagonist agent trains, since this is the relevant metric for protagonist training. Note that this means the compute necessary for training PBRL is



(a) Laser Tag

(b) Simple Push

Figure 10.2: Illustrations of the (a) Laser Tag and (b) Simple Push environments. Laser Tag is a symmetric gridworld game, where agents get points by “tagging” the other agent with a light beam. Simple Push is an asymmetric continuous control game, consisting of *aggressor* and *protector* agents. The protector knows which landmark is the *target* rather than the *decoy*. The protector wishes to be close to the *target* while keeping the aggressor away from it.

$n + 1$ times higher than self-play at the same number of training steps (although PBRL is more parallelizable).

We train all policies — self-play, PBRL, and adversarial—using proximal policy optimization [PPO; 144]. PPO is widely used and has achieved good results with self-play in complex environments [14]. Furthermore, PPO was used in Chapter 8 to train adversarial policies in similar continuous control environments. We use the PPO implementation in `RLlib` [92], from the `ray` library [109], due to its support for multi-agent environments and parallelizing RL training.

10.3 Experiments

We evaluate the PBRL defense in two low-dimensional environments, described in Section 10.3.1. In Section 10.3.2, we confirm that baseline self-play policies are vulnerable to attack. To the best of our knowledge, these are the lowest-dimensional environments in which an adversarial policy has been found. Finally, in Section 10.3.3 we find that PBRL improves robustness against adversarial policies and explore the relationship with population size.

Unless otherwise noted, in all experiments we train 5 seeds of defender policies. We attack each defender using 3 seeds of adversaries for a total of 15 adversaries. Unless omitted for legibility, 95% confidence intervals are shown as shaded regions for training curves and bars in bar plots.

10.3.1 Environments

We evaluate in two low-dimensional, two-player zero-sum games illustrated in Figure 10.2: *Laser Tag* and *Simple Push*. We choose relatively simple environments compared to Chapters 8 and 9 as even self-play in those environments would be at the limits of our computational resources, and the PBRL defense further inflates the computational requirements.

Laser Tag is a symmetric game with incomplete information [86]. The players see 17 spaces in front, 10 to the sides, and 2 spaces behind their agent. The two agents move on a grid world and get points for tagging each other with a light beam. Obstacles block movement and beams. We make the environment zero-sum by also subtracting a point from the tagged player.

Simple Push is a continuous environment introduced by Mordatch and Abbeel [106] and released with Lowe et al. [96]. The environment is asymmetric, with one agent the *aggressor* and the other the *protector*.[*]

The environment contains two randomly placed landmarks. Only the protector knows which of these is the true target; the other landmark acts as a "decoy" for the aggressor. The aggressor receives positive reward based on the protector's distance to the true target. Subtracted from this is a relative penalty, based on its own distance. Unlike vanilla *Simple Push*, where the protector's rewards are solely based on its own distance, we make the environment zero-sum by giving the protector the negative of the aggressor's reward.

As *Simple Push* is very low dimensional (a two-dimensional continuous control task), we develop a variant with a "cheap talk" communication channel (see Section F.1 in the appendix) that increases the dimensionality but does not otherwise change the dynamics. This channel extends the action and observation spaces allowing agents to send one-hot coded tokens to each other. See Section F.1 of the appendix for details on the communication channel.

10.3.2 Self-play baseline

Before evaluating our PBRL defense, we first consider the exploitability of the self-play baseline using the attack from Chapter 8. We find self-play policies to be vulnerable in *Simple Push* with a communication channel. In *Laser Tag*, adversarial policies can be found, however variance is high and some self-play defenders are hard to attack.

Initial experiments, whose training curves can be found in Figure F.1 of the appendix, showed that the attack fails in vanilla *Simple Push.* Consequently, we performed subsequent experiments in *Simple Push* with a one-hot coded communication channel of 50 tokens. These results add nuance to the finding from Chapter 8 that adversarial policies are easier to find in higher-dimensional environments. The environments in this chapter are significantly lower-dimensional than those considered in Chapter 8, suggesting the minimum dimensionality for attack is fairy small. However, the fact that policies are only vulnerable in *Simple Push* given

[*] Note, that the notion of *aggressor* and *protector* in this environment is orthogonal to *adversary* and *defender* in the sense of adversarial policies.

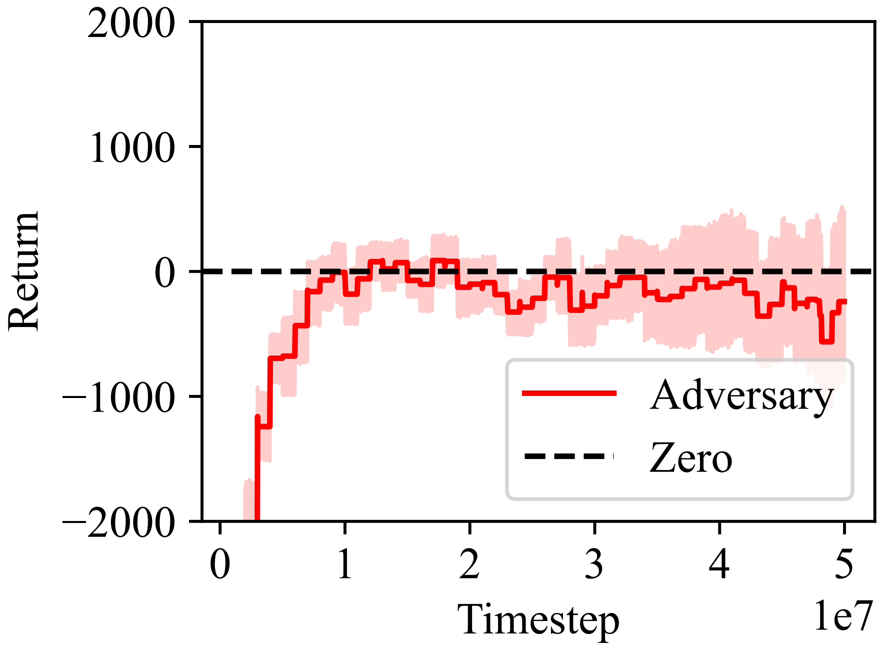


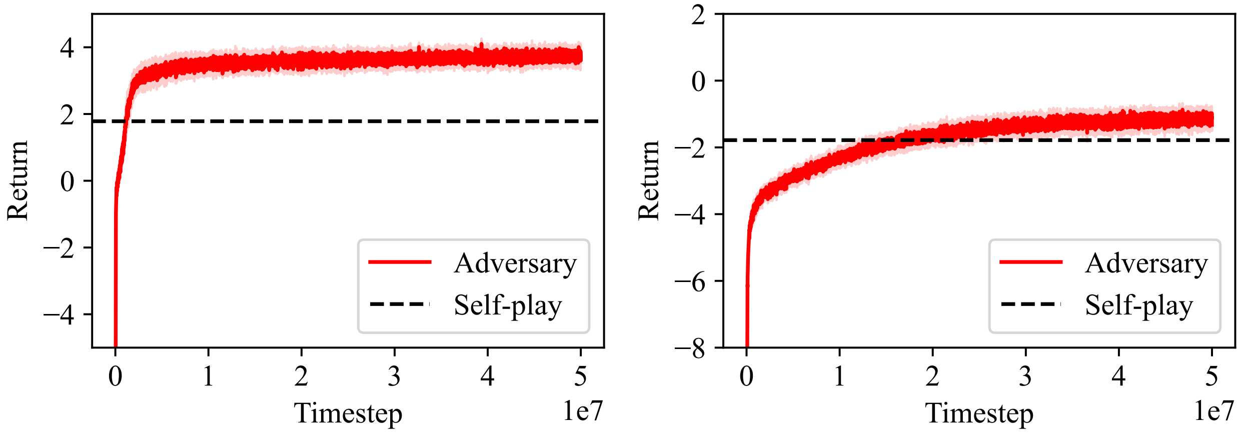
Figure 10.3: Training curve of adversaries in *Laser Tag* against the self-play baseline. An adversarial policy can on average be found after fewer than 15 million timesteps of training. However the variance between seeds is high, and adversary performance deteriorates against some defenders over time.

a communication channel, and the high variance in the defender robustness in *Laser Tag*, supports the previous claim that dimensionality is an important mediator for exploitability.

Laser Tag. We train self-play policies in the symmetric *Laser Tag* environment for 25 million timesteps. This should be adequate to produce a strong policy, as the paper introducing the environment [86] trained self-play for only 3 million timesteps. Figure 10.3 shows the average return of the adversaries trained against these defenders. We train adversaries for 50 million timesteps, twice as many as the defenders, in order to reason about the adversaries' behavior given more compute. Since the game is symmetric, an agent with a return above zero outperforms its opponent.

Successful adversarial policies can be found within, on average, fewer than 15 million timesteps. The loose confidence interval suggests high variability between different seeds: some of the trained defenders are robust while others are not. On average, attacker performance deteriorates after 20 million timesteps, which suggests an instability in adversary training.

Simple Push. Since *Simple Push* has an asymmetric observation space, we train self-play using a separate policy for each player. We train the agents for 25 million timesteps, which



(a) Defender return.

(b) Aggressor return.

Figure 10.4: Training curve of adversaries in *Simple Push* against the self-play baseline, where the adversary controls the aggressor (**left**) or defender (**right**). The environment includes a 50-token communication channel. The black dotted line marks the return achieved by the self-play training baseline controlling the respective agent at the end of training. The attack substantially outperforms the self-play baseline when the adversary controls the protector (right), but only barely outperforms the baseline when the adversary controls the aggressor (left).

we expect to be more than sufficient given the 625,000 timesteps used in prior work [96]. While prior work used the multi-agent deep deterministic policy gradient (MADDPG), not PPO, in exploratory experiments we find PPO to perform comparably to MADDPG.

Again we train adversaries for 50 million timesteps, twice as many as the defenders, in order to reason about the adversaries' behavior given more compute. Figure 10.4a shows the when the adversary controls the protector, the adversary achieves almost twice the return on average (in red) as the self-play baseline (in black) at 25 million timesteps, which is the point where adversary and defender trained for the same number of timesteps. By contrast, Figure 10.4b shows that when controlling the aggressor, the adversary needs around 20 million timesteps just to match the defender. Return after training the adversary for twice as many timesteps as the defender is only slightly higher than the baseline.

These results suggest that protector policies in *Simple Push* are more robust to adversarial policies than aggressor policies are. We conjecture this asymmetry is due to the protector having more information than the aggressor: it knows the target landmark. Consequently, the aggressor needs to observe the opponent to learn the true target landmark. The protector could exploit this and perform movements that fool a defender aggressor. Due to the stronger nature of the attack controlling the protector, we focus our defense in the upcoming section on adversaries that control the protector aggent.

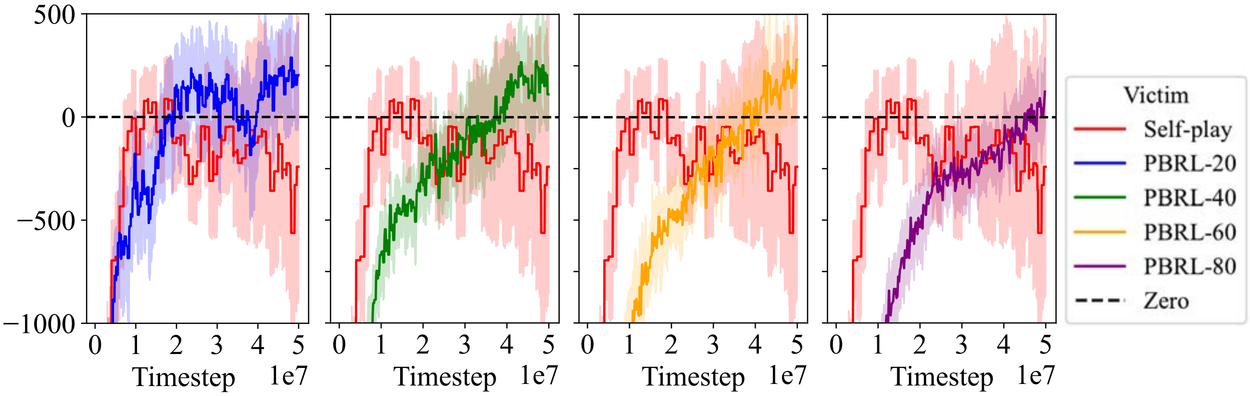


Figure 10.5: Training curves of adversaries in *Laser Tag* against defenders trained with PBRL against (from left to right) 20, 40, 60, and 80 opponents. We measure exploitability by how few timesteps the adversary needs to train until outperforming the defender (i.e., achieving larger than 0 return). We find that exploitability decreases with increasing PBRL population size. Notably, self-play is exploitable within 15 million timesteps whereas PBRL-80 (right) is only exploitable after 40 million timesteps. We observe significant instability in the training of the adversary, with performance against the self-play baseline beginning to *decline* after 15 million timesteps, with similar instability present after 30 million timesteps against PBRL-20.

10.3.3 PBRL defense

In this section, we evaluate the effectiveness of our PBRL defense by trying to exploit PBRL-trained policies. We train our adversaries for up to 50 million timesteps against the same fixed 25-million-timestep defender to evaluate how many timesteps are needed to attack more robust defenders. We find some improvement in robustness relative to the self-play baseline in both environments. In *Laser Tag*, larger populations increase the number of timesteps needed to find the first adversarial policy — at the cost of requiring more computational resources. In *Simple Push*, we find that PBRL significantly outperforms the self-play baseline, with as few as $n = 2$ opponents being sufficient.

Laser Tag. To explore the impact of population size, we train policies with $n = 20$, 40, 60, and 80 opponents in *Laser Tag*. Figure 10.5 shows the average return of adversaries attacking these hardened protagonists. We find that using PBRL increases robustness: finding an adversary that achieves higher than 0 reward takes more timesteps on average. While fairly noisy, generally the number of timesteps needed to outperform the defender — when return exceeds the zero line — grows with increasing population size. An adversary attacking a protagonist trained against a population of size 80 needs to train for almost double the timesteps as the self-play defender. However, there are diminishing returns to population size, with the relative difference in timesteps growing smaller.

Although the adversarial policy was trained for up to double the number of timesteps as the protagonist agent, note that PBRL used 80 times as much compute for every timestep of the protagonist, since it had to train the opponents for the same number of timesteps. Additionally, we find that adversaries that continue to train eventually do outperform PBRL defenders on average, whereas average performance against self-play defenders *decreases* over time. The 95% confidence intervals in Figure 10.5 show that the variance of adversaries attacking self-play increases over time, which suggests adversary training to be less stable when attacking self-play as opposed to attacking PBRL. Self-play seems to converge to policies of widely varying robustness, whereas in PBRL the variance of the adversary's return is lower.

A possible explanation for why the defender's are vulnerable to advesrarial attack is that the fairly small neural networks used in RL may be unable to represent more robust policies. However, our results show improvements in protagonist performance as population size increases, even while holding the model architecture fixed. This suggests that a lack of model capacity is not the main cause of adversarial policies success.

Simple Push. We focus on making the *aggressor* agent more robust in *Simple Push*, as Section 10.3.2 showed that the defender self-play policy is already relatively robust to attack. Consequently, the protagonist controls the *aggressor* and adversaries control the *protector* agent. We use PBRL to train against n = 2, 4, 8, and 16 opponents. Since the environment is not balanced, as a baseline we compare the adversary's performance to those of the average return achieved by the PBRL opponents at the end of training against the same defender policy.

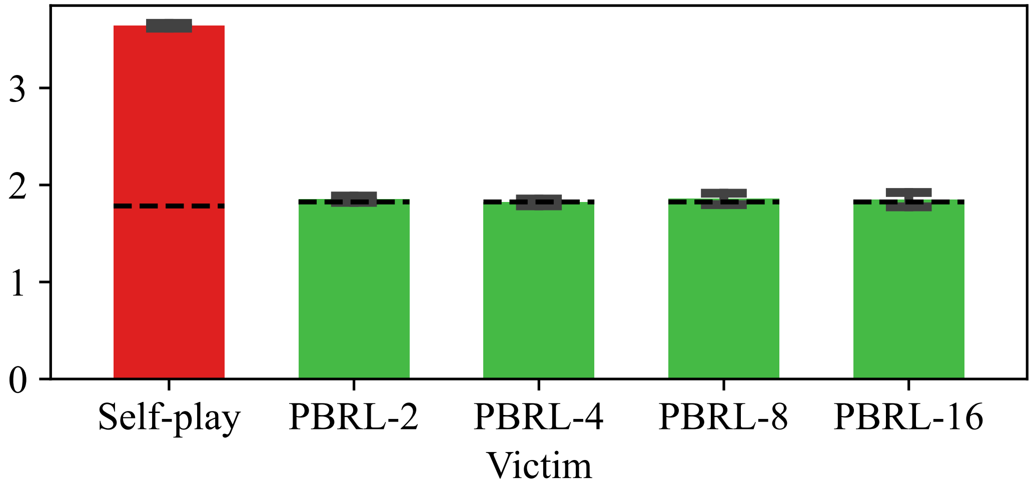


Figure 10.6: Adversary return (y-axis) against defenders (x-axis) in *Simple Push*. PBRL-trained defenders (green) are significantly more robust to adversarial attack than the self-play baseline (red) and attain similar return to the baseline that plays against a *non-adversarial* agent (black dotted line).

Figure 10.6 shows the returns at 25 million timesteps. Since baselines in different setups could converge to different returns, we calculate separate baseline thresholds for each of the 4 settings (in addition to the self-play baseline from Figure 10.4), marked by the dashed black line. The PBRL-trained agent is significantly more robust than the self-play policy, in red. In fact, the PBRL protagonist achieves similar return *under a zero-shot attack* as the self-play policy does *against its self-play opponent* (the dashed threshold). Notably, the values all PBRL policies converge to differ by less than 3%.

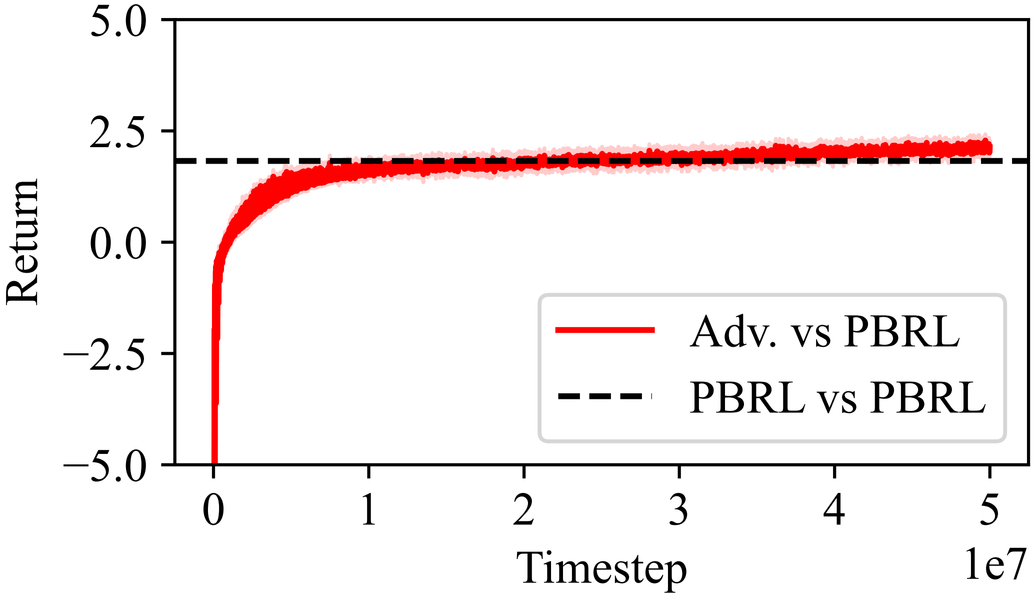


Figure 10.7: Adversary return (y-axis) over time in *Simple Push*, averaged over 60 PBRL-trained defender agents (number of opponents $n \in \{2, 4, 8, 16\}$). The tight confidence interval shows that defenders are all similarly robust in *Simple Push*. The adversarial return only crosses the zero line around 25 million timesteps, indicating adversaries need at least as many timesteps as the defender in order to outperform it.

Figure 10.7 shows the training curve when training adversaries for up to 50 million timesteps. Since there is no discernible difference in the 4 PBRL settings, we average over these for a total of 60 adversarial policies. The attack eventually outperforms the PBRL protagonist defender, but is only slightly stronger even after training for *twice as long.*

Although PBRL is significantly more robust than self-play (effectively PBRL with $n = 1$ opponent), perhaps surprisingly there is little benefit from using more than $n = 2$ opponents. In particular, there is no clear decrease in robustness when using n as low as 2, which is the lowest PBRL setting that does not degenerate to self-play. This is in contrast to *Laser Tag*, which saw large differences in robustness depending on population size.

This difference is likely due to *Simple Push* having only a handful of high-level strategies that can be pursued. By contrast, the *Laser Tag* environment allows for more variation in the details of possible strategies, making it harder to achieve full robustness. Additionally, it is possible that the intervention of slightly increasing dimensionality by adding a "cheap talk" channel can be circumvented with minimally higher diversity during training.

10.4 Limitations and future work

We have shown that our hardened PBRL defender agents are significantly more robust, often only being exploitable when the adversary is trained for more timesteps than the protagonist defender. However, PBRL is computationally very demanding, since it *also* requires training n opponents for as many timesteps as the protagonist. Accordingly, adversaries can still exploit defenders given the same compute and training timestep budget.

However, in many cases defenders have access to more compute and training timesteps. First, defenders may be able to limit the number of timesteps an attacker can use to train against the defender, such as if access to the policy is behind a rate-limited API. Second, if a small number of opponents suffices (such as $n = 2$ in *Simple Push*) then the defender only needs to have *slightly* more compute than the attacker. Finally, defenders often have significant computational resources: although PBRL is unlikely to prevent an attack from a sophisticated adversary like a nation state, it may be enough to defeat many low-resource attacks. Nonetheless, reducing this computational overhead is an important direction for future work. For example, can we obtain similar performance with fewer opponents if we train them to be maximally diverse from one another?

In addition, *if* additional compute resources are available, our approach allows a defender to *make use* of them. Once an agent has converged, using additional compute to continue training in self-play is usually of no use. However, convergence is not sufficient for robustness — as illustrated by the existence of adversarial policies. Our approach enables a purposeful use for additional computing power.

A key open question is how the number of opponents n required for robustness scales with the complexity of the environment. PBRL will scale poorly if the required population size is proportional to the size of the state space: in more complex environments each opponent will take longer to train *and* more opponents will be required. But a priori it seems likely that n may depend more on the number of high-level strategies in the environment. This is only loosely related to the dimensionality of the state space. For example, some simple matrix games have high strategic complexity, while some high-dimensional video games have only a handful of sensible strategies.

Our evaluation uses the same adversarial policy attack introduced in Chapter 8, which we established was strong enough to exploit unhardened defenders in these environments. However, it is possible that alternative attacks would be able to exploit even our hardened defender. We hope to see iterative development of stronger attacks and defenses, similar to the trend in adversarial examples more broadly.

10.5 Conclusion

We introduced the defense by PBRL as a method to reduce exploitability of RL policies. Our results show an increase in zero-shot robustness against new adversaries compared to self-play training. However, some self-play defenders are naturally robust, and PBRL comes with an increased computational cost. We find that the size of the population necessary depends on the environment used, and larger populations can increase overall robustness. This work suggests that increasing diversity during training can lead to improved robustness and contributes toward the goal of making agents less exploitable. Source code is available at `https://github.com/HumanCompatibleAI/reducing-exploitability`.

Chapter 11

Conclusion

Machine learning has the potential to be enormously beneficial to humanity, whether by automating mundane tasks or by performing certain tasks better than a human could. However, existing techniques are insufficient to ensure machine learning systems reliably perform open-ended tasks, especially in safety-critical domains. Until this is resolved, machine learning systems can only be safely deployed in limited domains where task specification is simple and occasional failure is tolerable. Overcoming these hurdles and building trustworthy machine learning is therefore a key challenge for the machine learning research community.

In this dissertation, we have decomposed trustworthy machine learning into two parts. First, the agent must learn an objective that is aligned with the goals of the human principal. This is often challenging, especially in complex, value-laden, open-ended tasks. Second, the agent must reliably and robustly optimize that objective. Although we cannot expect the agent to perform well in arbitrary situations, performance should degrade gracefully as we go off-distribution, and we must avoid unpredictable catastrophic failures.

We have introduced several complementary methods for understanding and testing learned rewards. In Chapter 3, we characterized the fundamental limits of different reward learning data sources. Next, in Chapter 4 we introduced the EPIC distance between reward functions and showed in Chapter 5 that this distance bounds the difference in returns of optimal policies. Finally, in Chapter 6 we developed an interpretability method to explain learned reward functions, taking into account the special structure of reward functions. Together, these methods provide a toolkit to evaluate learned reward functions both for practitioners seeking to learn rewards and for researchers seeking to benchmark new reward learning algorithms.

Specifying the objective correctly is an important component of trustworthy machine learning, but it is not sufficient. The agent must also reliably pursue the objective across the wide range of settings that may be experienced during deployment in real-world scenarios. We studied reliability through the lens of adversarial robustness, introducing in Chapter 7 a novel physically realistic threat model for adversaries in sequential decision-making. In Chapters 8 and 9, we found that state-of-the-art and even human-level AI systems are vulnerable to this attack, failing in surprising ways. This shows that even seemingly highly capable systems may contain hidden failure modes, emphasizing the importance of testing using approaches such as these. Finally, in Chapter 10 we showed that training can be made more robust using population-based RL.

11.1 Limitations and future work

Infinite-data assumption Our analysis of reward learning data sources in Chapter 3 considers their informativeness in the infinite-sample limit. Although this provides a useful upper bound, for practical applications we care about their informativeness given a finite number of samples. We expect that some data sources, such as demonstrations, will be highly informative early on but have sharply diminishing returns, whereas other data sources, such as preference comparisons, may provide less information per sample but reach their asymptote at a higher level. We hope to explore this systematically, either theoretically or empirically, in future work.

Equivalence classes The EPIC distance introduced in Chapter 4 uses a conservative equivalence class for reward functions, consisting of just potential shaping and positive affine transformations, that is guaranteed to never change optimal policies in any MDP. However, in practice we often have additional information about the structure of an MDP; for example, we know that energy is conserved in a physical system, even if we may not know the details of all relevant physical parameters such as the exact friction at a joint. The set of optimal-policy-preserving transformations under such assumptions is broader, and we could get more informative distances if we include those in the equivalence class.

Wulfe et al. [179] provide an initial step in this direction by adapting EPIC to use the transition dynamics. Unfortunately, however, this modification loses some of EPIC's desirable properties, such as the canonicalization remaining within the equivalence class. Moreover, their method requires knowing the transition dynamics exactly, whereas in general we might only know some invariants. An important direction for future work would be to incorporate such invariances while still preserving the theoretical desiderata of EPIC.

Generality of adversarial policies We have found adversarial policies in state-of-the-art continuous control policies (Chapter 8) and in professional-level Go-playing AI systems (Chapter 9). Although this provides significant evidence that adversarial policies are a fairly common phenomenon in self-play policies, we still do not know whether they are ubiquitous or merely commonly occurring. We plan to attempt to attack other highly capable AI systems, such as Leela Chess [94] and Polygames for Hex [32]. If these other systems are similarly exploitable, this would provide strong evidence that self-play in general fails to produce robust policies.

Understanding adversarial policies Our study of adversarial policies and their effect on the defender indicates that they win by fooling the defender, likely in part by inducing highly off-distribution inputs and, therefore, activations in the defender's network. We conjecture that defenders are vulnerable to adversarial policies because self-play may converge to local equilibria. This is supported by our finding in Chapter 10 that population-based RL makes defenders harder to exploit. However, we currently lack a rigorous understanding of how and why adversarial policies emerge. Further study here is warranted as it may provide general insights about the limitations of self-play, deep learning, and related algorithms, and suggest avenues for improvement.

Computationally efficient defenses The population-based RL defense in Chapter 10 is promising, but the computational overhead introduced makes it challenging to deploy for complex games, where even self-play can be extremely computationally demanding. We plan to investigate ways to reduce the computational overhead, such as by using a smaller population of opponents that is explicitly optimized for diversity. When the computational overhead is reduced, we intend to evaluate the defense in more strategically complex environments such as Go.

11.2 Closing thoughts

The importance of machine learning systems being trustworthy is widely acknowledged. However, the results in this dissertation show that it can be far from obvious whether or not a system can be trusted. We have seen that many learned reward functions can be very fragile, despite having high predictive accuracy on the training data. Moreover, even highly capable AI systems that have been widely studied and that beat top professionals can fall prey to exploits that would not fool even a human amateur. These results demonstrate that AI systems that reach human performance may still have alien goals and internal representations that can produce surprising and often undesirable behavior.

We hope that the methodologies developed in this dissertation help researchers and practitioners build trustworthy ML systems and validate if their systems are trustworthy prior to deployment. Although there remains significant scope for improvement, we believe these techniques are already a valuable part of the toolkit and could grow to become staples of training and evaluation. However, it would be a mistake to rely on these or indeed any set of technical tools: we would argue the ubiquity of difficult-to-detect and potentially catastrophic failure modes also requires a shift in *culture*. In particular, the machine learning community could borrow many of the mental tools and organizational structures already found to be effective in the computer security and safety engineering communities, which have long tackled similar problems [89].

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, pp. 1–8.

[2] Martin Aigner. *Combinatorial Theory*. Reprint of the 1979. Classics in Mathematics. Springer Berlin Heidelberg, 1996.

[3] Riad Akrour, Marc Schoenauer, and Michèle Sebag. “APRIL: Active Preference Learning-Based Reinforcement Learning”. In: *Machine Learning and Knowledge Discovery in Databases: ECML PKDD 2012, Proceedings, Part II*. Vol. 7524. Lecture Notes in Computer Science. Bristol, UK: Springer, 2012, pp. 116–131.

[4] Riad Akrour, Marc Schoenauer, and Michele Sebag. “Preference-Based Policy Learning”. In: *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2011*. Vol. 6911. Lecture Notes in Computer Science. Berlin: Springer, 2011.

[5] Louis Victor Allis. “Searching for Solutions in Games and Artificial Intelligence”. PhD thesis. Maastricht University, 1994.

[6] Kareem Amin, Nan Jiang, and Satinder P. Singh. “Repeated Inverse Reinforcement Learning”. In: *Proceedings of the 31st Conference on Neural Information Processing Systems*. Vol. 30. 2017, pp. 1813–1822.

[7] Dario Amodei, Paul Christiano, and Alex Ray. *Learning from Human Preferences*. June 2017. URL: `https://openai.com/blog/deep-reinforcement-learning-from-human-preferences/`.

[8] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. *Concrete Problems in AI Safety*. 2016. arXiv: `1606.06565 [cs.AI]`.

[9] Stuart Armstrong and Sören Mindermann. “Occam’s Razor is Insufficient to Infer the Preferences of Irrational Agents”. In: *Proceedings of the 32nd Conference on Neural Information Processing Systems*. Vol. 31. 2018, pp. 5603–5614.

[10] J. Andrew Bagnell, Andrew Y. Ng, and Jeff G. Schneider. *Solving Uncertain Markov Decision Processes*. Tech. rep. CMU-RI-TR-01-25. Aug. 2001.

[11] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. "Learning to Understand Goal Specifications by Modelling Reward". In: *Proceedings of the Seventh International Conference on Learning Representations (ICLR)*. 2019.

[12] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback*. 2022. arXiv: `2204.05862`.

[13] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. "Open-ended learning in symmetric zero-sum games". In: *Proceedings of the Thirty-sixth International Conference on Machine Learning (ICML)*. 2019, pp. 434–443.

[14] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. "Emergent Complexity via Multi-Agent Competition". In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. 2018.

[15] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. *Source code and model weights for Emergent Complexity via Multi-Agent Competition*. 2018. URL: `https://github.com/openai/multiagent-competition`.

[16] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. "Verifiable Reinforcement Learning via Policy Extraction". In: *Advances in Neural Information Processing Systems (NeurIPS) 31*. 2018, pp. 2494–2504.

[17] Petr Baudiš and Jean-loup Gailly. *PACHI Readme*. 2020. URL: `https://github.com/pasky/pachi/blob/a7c60ec10e1a071a8ac7fc51f7ccd62f006fff21/README.md` (visited on 11/08/2022).

[18] Petr Baudiš and Jean-loup Gailly. "PACHI: State of the Art Open Source Go Program". In: *Advances in Computer Games: 13th International Conference, ACG 2011*. Vol. 7168. Lecture Notes in Computer Science. 2012, pp. 24–38.

[19] Vahid Behzadan and Arslan Munir. "Adversarial Reinforcement Learning Framework for Benchmarking Collision Avoidance Mechanisms in Autonomous Vehicles". In: *IEEE Intelligent Transportation Systems Magazine* 13.2 (2021).

[20] David B. Benson. "Life in the Game of Go". In: *Information Sciences* 10.1 (1976), pp. 17–29.

[21] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. *Dota 2 with Large Scale Deep Reinforcement Learning.* 2019. arXiv: `1912.06680 [cs.LG]`.

[22] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. "Learning Reward Functions From Diverse Sources of Human Feedback: Optimally Integrating Demonstrations and Preferences". In: *The International Journal of Robotics Research* 41.1 (2022), pp. 45–67.

[23] Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies.* Oxford: Oxford University Press, 2014.

[24] Justin Boyan and Andrew W Moore. "Generalization in Reinforcement Learning: Safely Approximating the Value Function". In: *Advances in Neural Information Processing Systems (NIPS).* 1994, pp. 369–376.

[25] Andries E. Brouwer. *Length of a Go Game.* 2014. URL: `https://homepages.cwi.nl/~aeb/go/misc/gostat.html` (visited on 09/27/2022).

[26] George W Brown. "Iterative Solution of Games by Fictitious Play". In: *Activity Analysis of Production and Allocation: Proceedings of a Conference.* 1951, p. 376.

[27] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. "Deep Counterfactual Regret Minimization". In: *Proceedings of the 36th International Conference on Machine Learning (ICML).* 2019, pp. 793–802.

[28] Noam Brown and Tuomas Sandholm. "Superhuman Ai for Heads-Up No-Limit Poker: Libratus Beats Top Professionals". In: *Science* 359.6374 (2018), pp. 418–424.

[29] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. *Scaling Data-Driven Robotics with Reward Sketching and Batch Reinforcement Learning.* 2019. arXiv: `1909.12200 [cs.RO]`.

[30] Haoyang Cao, Samuel N. Cohen, and Lukasz Szpruch. "Identifiability in Inverse Reinforcement Learning". In: *Proceedings of the 35th Conference on Neural Information Processing Systems.* Vol. 34. 2021, pp. 12362–12373.

[31] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. *On Evaluating Adversarial Robustness.* arXiv:1902.06705v2 [cs.LG]. 2019. arXiv: `1902.06705`.

[32] Tristan Cazenave, Yen-Chi Chen, Guan-Wei Chen, Shi-Yu Chen, Xian-Dong Chiu, Julien Dehos, Maria Elsa, Qucheng Gong, Hengyuan Hu, Vasil Khalidov, et al. "Polygames: Improved Zero Learning". In: *ICGA Journal* 42.4 (2020), pp. 244–256.

[33] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. “Deep Reinforcement Learning from Human Preferences”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 4299–4307.

[34] Anne G E Collins and Amitai Shenhav. “Advances in Modeling Learning and Decision-Making in Neuroscience”. In: *Neuropsychopharmacology* 47 (2022), pp. 104–118.

[35] Rémi Coulom. *Go Ratings*. 2022. URL: `https://archive.ph/H0VDl` (visited on 09/28/2022).

[36] Wojciech M. Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. “Real World Games Look Like Spinning Tops”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[37] Pavel Czempin and Adam Gleave. “Reducing Exploitability with Population Based Training”. In: *ICML Workshop on New Frontiers in Adversarial Machine Learning*. 2022.

[38] Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. “Distributional Reinforcement Learning with Quantile Regression”. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. Vol. 32(1). 2018, pp. 2892–2901.

[39] Grégoire Déletang, Jordi Grau-Moya, Miljan Martic, Tim Genewein, Tom McGrath, Vladimir Mikulik, Markus Kunesch, Shane Legg, and Pedro A. Ortega. *Causal Analysis of Agent Behavior for AI Safety*. 2021. arXiv: `2103.03938 [cs.AI]`.

[40] Daniel C Dennett. *The Intentional Stance*. Cambridge, MA: MIT Press, 1987.

[41] Rati Devidze, Goran Radanovic, Parameswaran Kamalaruban, and Adish Singla. “Explicable Reward Design for Reinforcement Learning Agents”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[42] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*. 2017, pp. 1–16.

[43] John Doyle, James A. Primbs, Benjamin Shapiro, and Vesna Nevistić. “Nonlinear Games: Examples and Counterexamples”. In: *Proceedings of 35th IEEE Conference on Decision and Control*. Vol. 4. 1996, pp. 3915–3920.

[44] Krishnamurthy Dvijotham and Emanuel Todorov. “Inverse Optimal Control with Linearly-Solvable MDPs”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. 2010, pp. 335–342.

[45] EGD. *European Go Database*. 2022. URL: `https://www.europeangodatabase.eu/EGD/` (visited on 09/28/2022).

[46] European Go Federation. *European Pros*. 2022. URL: `https://www.eurogofed.org/pros/` (visited on 11/08/2022).

[47] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control Via Policy Optimization". In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2016, pp. 49–58.

[48] Peter C Fishburn. *Utility Theory for Decision Making*. New York: John Wiley & Sons, Inc., 1970.

[49] Justin Fu, Katie Luo, and Sergey Levine. "Learning Robust Rewards with Adverserial Inverse Reinforcement Learning". In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. 2018.

[50] Iason Gabriel. "Artificial Intelligence, Values, and Alignment". In: *Minds and Machines* 30.3 (2020), pp. 411–437.

[51] Shromona Ghosh, Felix Berkenkamp, Gireeja Ranade, Shaz Qadeer, and Ashish Kapoor. "Verifying Controllers Against Adversarial Examples with Bayesian Optimization". In: *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 7306–7313.

[52] Justin Gilmer, Ryan P. Adams, Ian Goodfellow, David Andersen, and George E. Dahl. *Motivating the Rules of the Game for Adversarial Example Research*. 2018. arXiv: 1807.06732v2 [cs.LG].

[53] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. *Adversarial Spheres*. 2018. arXiv: 1801.02774 [cs.CV].

[54] Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, and Jan Leike. "Quantifying Differences in Reward Functions". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2021.

[55] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. "Adversarial Policies: Attacking Deep Reinforcement Learning". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2020.

[56] I.J. Good. "On the Principle of Total Evidence". In: *The British Journal for the Philosophy of Science* 17.4 (1967), pp. 319–321.

[57] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, Pieter Abbeel, and Jack Clark. *Attacking Machine Learning with Adversarial Examples*. 2017. URL: https://openai.com/blog/adversarial-example-research/.

[58] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.

[59] C. A. E. Goodhart. "Problems of Monetary Management: The UK Experience". In: *Monetary Theory and Practice*. Springer, 1984, pp. 91–121.

[60] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. "Reinforcement Learning with Deep Energy-Based Policies". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 1352–1361.

[61] Dylan Hadfield-Menell and Gillian K. Hadfield. "Incomplete Contracting and AI Alignment". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (AIES)*. 2019, pp. 417–422.

[62] Fu Haoda and David J. Wu. *summarize_sgfs.py*. 2022. URL: `https://github.com/lightvector/KataGo/blob/c957055e020fe438024ddffd7c5b51b349e86dcc/python/summarize_sgfs.py` (visited on 09/28/2022).

[63] Johannes Heinrich, Marc Lanctot, and David Silver. "Fictitious Self-Play in Extensive-Form Games". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 2015, pp. 805–813.

[64] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. "Natural Adversarial Examples". In: *CVPR*. June 2021.

[65] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. *Stable Baselines*. `https://github.com/hill-a/stable-baselines`. 2018.

[66] Andrew Howes, Richard L Lewis, and Satinder Singh. "Utility Maximization and Bounds on Human Information Processing". In: *Topics in Cognitive Science* 6.2 (2014), pp. 198–203.

[67] Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. *Adversarial Attacks on Neural Network Policies*. 2017. arXiv: `1702.02284 [cs.LG]`.

[68] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. "Reward Learning from Human Preferences and Demonstrations in Atari". In: *Proceedings of the 32nd Conference on Neural Information Processing Systems*. Vol. 31. 2018, pp. 8022–8034.

[69] Inaam Ilahi, Muhammad Usama, Junaid Qadir, Muhammad Umar Janjua, Ala Al-Fuqaha, Dinh Thai Hoang, and Dusit Niyato. "Challenges and Countermeasures for Adversarial Attacks on Deep Reinforcement Learning". In: *IEEE TAI* 3.2 (2022), pp. 90–109.

[70] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. *Adversarial Examples Are Not Bugs, They Are Features*. 2019. arXiv: `1905.02175v4 [stat.ML]`.

[71] Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. "Human-level Performance in 3d Multiplayer Games with Population-Based Reinforcement Learning". In: *Science* 364.6443 (May 31, 2019), pp. 859–865.

[72] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. *Population Based Training of Neural Networks*. 2017. arXiv: `1711.09846 [cs.LG]`.

[73] Erik Jenner and Adam Gleave. *Preprocessing Reward Functions for Interpretability*. 2022. arXiv: `2203.13553 [cs.LG]`.

[74] Hong Jun Jeon, Smitha Milli, and Anca Dragan. "Reward-Rational (Implicit) Choice: A Unifying Formalism for Reward Learning". In: *Proceedings of the 34th Conference on Neural Information Processing Systems*. Vol. 33. 2020, pp. 4415–4426.

[75] Michael Johanson, Kevin Waugh, Michael H. Bowling, and Martin Zinkevich. "Accelerating Best Response Calculation in Large Extensive Games". In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

[76] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. "Explainable Reinforcement Learning via Reward Decomposition". In: *IJCAI Workshop on Explainable Artificial Intelligence (XAI)*. 2019, pp. 47–53.

[77] KGS. *gnugo2 Rank Graph*. 2022. URL: `https://www.gokgs.com/graphPage.jsp?user=gnugo2` (visited on 11/08/2022).

[78] KGS. *Top 100 KGS Players*. 2022. URL: `https://archive.ph/BbAHB` (visited on 09/26/2022).

[79] Marc Khoury and Dylan Hadfield-Menell. *On the Geometry of Adversarial Examples*. 2018. arXiv: `1811.00525 [cs.LG]`.

[80] Kuno Kim, Shivam Garg, Kirankumar Shiragur, and Stefano Ermon. "Reward Identification in Inverse Reinforcement Learning". In: *Proceedings of the 38th International Conference on Machine Learning*. 2021, pp. 5496–5505.

[81] Lora Kolodny. *Cruise Gets Green Light for Commercial Robotaxi Service in San Francisco*. URL: `https://www.cnbc.com/2022/06/02/cruise-gets-green-light-for-commercial-robotaxis-in-san-francisco.html`.

[82] Pallavi Koppol, Henny Admoni, and Reid Simmons. "Iterative Interactive Reward Learning". In: *Participatory Approaches to Machine Learning, International Conference on Machine Learning (ICML) Workshop*. 2020.

[83] Jernej Kos and Dawn Song. *Delving Into Adversarial Attacks on Deep Policies*. 2017. arXiv: `1705.06452 [stat.ML]`.

[84] Sampo Kuutti, Saber Fallah, and Richard Bowden. “ARC: Adversarially Robust Control Policies for Autonomous Vehicles”. In: *Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 522–529.

[85] Li-Cheng Lan, Huan Zhang, Ti-Rong Wu, Meng-Yu Tsai, I-Chen Wu, and Cho-Jui Hsieh. “Are AlphaZero-like Agents Robust to Adversarial Perturbations?” In: *Advances in Neural Information Processing Systems*. 2022.

[86] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. “A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017, pp. 4190–4203.

[87] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Philadelphia: SIAM, 1995.

[88] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. *Scalable Agent Alignment via Reward Modeling: A Research Direction*. 2018. arXiv: `1811.07871 [cs.LG]`.

[89] Nancy G. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA: MIT Press, Jan. 2012.

[90] Arthur Lewbel. “The Identification Zoo: Meanings of Identification in Econometrics”. In: *Journal of Economic Literature* 57.4 (2019), pp. 835–903.

[91] Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. “Deal or No Deal? End-to-End Learning of Negotiation Dialogues”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.

[92] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. “RLlib: Abstractions for Distributed Reinforcement Learning”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 3053–3062.

[93] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. “Tactics of Adversarial Attack on Deep Reinforcement Learning Agents”. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 3756–3762.

[94] Gary Linscott, Alexander Lyashuk, François Pays, and other open-source contributors. *Leela Chess Zero*. 2022. (Visited on 10/16/2022).

[95] Viliam Lisý and Michael Bowling. “Equilibrium Approximation Quality of Current No-Limit Poker Bots”. In: *AAAI-17 Workshop on Computer Poker and Imperfect Information Game*. 2017. arXiv: `1612.07547 [cs.GT]`.

[96] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 6379–6390.

[97] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[98] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. 2018.

[99] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. "Explainable Reinforcement Learning Through a Causal Lens". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 3. 2020.

[100] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese. "Adversarially Robust Policy Learning: Active Construction of Physically-Plausible Perturbations". In: *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 3932–3939.

[101] David Manheim and Scott Garrabrant. *Categorizing Variants of Goodhart's Law*. arXiv: 1803.04585v4 [cs.LG]. 2018. arXiv: `1803.04585 [cs.AI]`.

[102] Charles F Manski. *Identification Problems in the Social Sciences*. Cambridge, MA: Harvard University Press, 1995.

[103] Charles F Manski. *Partial Identification of Probability Distributions*. New York: Springer, 2003.

[104] Lev McKinney, Yawen Duan, David Krueger, and Adam Gleave. "On The Fragility of Learned Reward Functions". In: *Deep Reinforcement Learning Workshop at NeurIPS*. 2022.

[105] Eric J. Michaud, Adam Gleave, and Stuart Russell. *Understanding Learned Reward Functions*. NeurIPS Deep RL Workshop. 2020. arXiv: `2012.05862 [cs.LG]`.

[106] Igor Mordatch and Pieter Abbeel. "Emergence of Grounded Compositional Language in Multi-Agent Populations". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[107] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. "Nonparametric Return Distribution Approximation for Reinforcement Learning". In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. 2010, pp. 799–806.

[108] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. "Parametric Return Density Estimation for Reinforcement Learning". In: *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*. 2010, pp. 368–375.

[109] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. “Ray: A Distributed Framework for Emerging AI Applications”. In: *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation*. 2018, pp. 561–577.

[110] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. second revised. Princeton, NJ: Princeton University Press, 1947.

[111] Andrew Y Ng, Daishi Harada, and Stuart Russell. “Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping”. In: *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*. 1999, pp. 278–287.

[112] Andrew Y Ng and Stuart Russell. “Algorithms for Inverse Reinforcement Learning”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. 2000, pp. 663–670.

[113] Laura Noonan. “JPMorgan Develops Robot to Execute Trades”. In: *Financial Times* (July 2017).

[114] OpenAI. *OpenAI Five*. `https://blog.openai.com/openai-five/`. 2018.

[115] Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. “What Matters for Adversarial Imitation Learning?” In: *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*. Vol. 34. 2021, pp. 14656–14668.

[116] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. *Training Language Models to Follow Instructions with Human Feedback*. 2022. arXiv: `2203.02155 [cs.CL]`.

[117] Malayandi Palan, Gleb Shevchuk, Nicholas Charles Landolfi, and Dorsa Sadigh. “Learning Reward Functions by Integrating Human Demonstrations and Preferences”. In: *Proceedings of Robotics: Science and Systems*. 2019.

[118] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. “The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2022.

[119] John Pascutto. *Leela Zero*. 2019. URL: `https://zero.sjeng.org/` (visited on 06/16/2022).

[120] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. “Robust Deep Reinforcement Learning with Adversarial Attacks”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 2018, pp. 2040–2042.

[121] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3803–3810.

[122] Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, Georgios Piliouras, Marc Lanctot, and Karl Tuyls. "From Poincaré Recurrence to Convergence in Imperfect Information Games: Finding Equilibrium via Regularization". In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. Vol. 139. 2021, pp. 8525–8535.

[123] Julien Perolat, Bart de Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, and Karl Tuyls. *Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning.* arXiv: 2206.15378v1 [cs.AI]. 2022. arXiv: 2206.15378 [cs.AI].

[124] Joshua C Peterson, David D Bourgin, Mayank Agrawal, Daniel Reichman, and Thomas L Griffiths. "Using Large-Scale Experiments and Machine Learning to Discover Theories of Human Decision-Making". In: *Science* 372.6547 (2021), pp. 1209–1214.

[125] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V. Le. "Meta Pseudo Labels". In: *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021.

[126] Tien Ho-Phuoc. *CIFAR10 to Compare Visual Recognition Performance between Deep Neural Networks and Humans*. 2018. arXiv: 1811.07270 [cs.CV].

[127] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. "Robust Adversarial Reinforcement Learning". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 2817–2826.

[128] Dean Pomerleau. "RALPH: Rapidly Adapting Lateral Position Handler". In: *Proceedings of the Intelligent Vehicles Symposium*. 1995, pp. 506–511.

[129] Erika Puiutta and Eric M. S. P. Veith. "Explainable Reinforcement Learning: A Survey". In: *Machine Learning and Knowledge Extraction*. 2020.

[130] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence, eds. *Dataset Shift in Machine Learning*. Cambridge, MA: MIT Press, 2008.

[131] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning". In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*. 2007, pp. 2586–2591.

[132] Hadi Ravanbakhsh and Sriram Sankaranarayanan. “Robust Controller Synthesis of Switched Systems Using Counterexample Guided Framework”. In: *Proceedings of the 2016 International Conference on Embedded Software (EMSOFT)*. 2016, 8:1–8:10.

[133] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. “Adversarial Attacks and Defenses in Deep Learning”. In: *Engineering* 6.3 (2020), pp. 346–360.

[134] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144.

[135] Rob. *NeuralZ06 Bot Configuration Settings*. 2022. URL: `https://discord.com/channels/417022162348802048/583775968804732928/983781367747837962` (visited on 06/16/2022).

[136] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3 (Dec. 2015), pp. 211–252.

[137] Jacob Russell and Eugene Santos. “Explaining Reward Functions in Markov Decision Processes”. In: *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference*. 2019, pp. 56–61.

[138] Stuart Russell. “Learning Agents for Uncertain Environments (Extended Abstract)”. In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT)*. 1998, pp. 101–103.

[139] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. Upper Saddle River, NJ: Prentice Hall, 2009.

[140] John Rust. “Structural Estimation of Markov Decision Processes”. In: *Handbook of Econometrics*. Ed. by Robert F. Engle and Daniel L. McFadden. Vol. 4. Amsterdam: Elsevier, 1994. Chap. 51, pp. 3081–3143.

[141] Dorsa Sadigh, Anca D. Dragan, Shankar Sastry, and Sanjit A. Seshia. “Active Preference-Based Learning of Reward Functions”. In: *Robotics: Science and Systems XIII*. 2017.

[142] Martin Schmid, Matej Moravcik, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, Zach Holland, Elnaz Davoodi, Alden Christianson, and Michael Bowling. *Player of Games*. arXiv: 2112.03178v1 [cs.LG]. 2021. arXiv: `2112.03178 [cs.AI]`.

[143] Paul J. H. Schoemaker. “The Expected Utility Model: Its Variants, Purposes, Evidence and Limitations”. In: *Journal of Economic Literature* 20.2 (1982), pp. 529–563.

[144] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: `1707.06347 [cs.LG]`.

[145] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. “Are Adversarial Examples Inevitable?” In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[146] Rohin Shah, Noah Gundotra, Pieter Abbeel, and Anca Dragan. “On the Feasibility of Learning, Rather than Assuming, Human Biases for Reward Inference”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. PMLR, June 2019, pp. 5670–5679.

[147] Rohin Shah, Steven H. Wang, Cody Wild, Stephanie Milani, Anssi Kanervisto, Vinicius G. Goecks, Nicholas Waytowich, David Watkins-Valls, Bharat Prakash, Edmund Mills, Divyansh Garg, Alexander Fries, Alexandra Souly, Jun Shern Chan, Daniel del Castillo, and Tom Lieberum. “Retrospective on the 2021 MineRL BASALT Competition on Learning from Human Feedback”. In: *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*. 2022, pp. 259–272.

[148] Vaishaal Shankar, Rebecca Roelofs, Horia Mania, Alex Fang, Benjamin Recht, and Ludwig Schmidt. “Evaluating Machine Accuracy on ImageNet”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020, pp. 8634–8644.

[149] Lloyd S. Shapley. “Stochastic Games”. In: *PNAS* 39.10 (1953), pp. 1095–1100.

[150] Buck Shlegeris, Fabien Rogers, and Lawrence Chan. *Language Models Seem to Be Much Better Than Humans at Next-Token Prediction*. Aug. 2022. URL: `https://www.alignmentforum.org/posts/htrZrxduciZ5QaCjw/language-models-seem-to-be-much-better-than-humans-at-next`.

[151] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (2016), pp. 484–489.

[152] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. “A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play”. In: *Science* 362.6419 (2018), pp. 1140–1144.

[153] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. “Mastering the Game of Go Without Human Knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359.

[154] Satinder P. Singh and Richard C. Yee. “An Upper Bound on the Loss from Approximate Optimal-Value Functions”. In: *Machine Learning* 16.3 (1994), pp. 227–233.

[155] Joar Skalse, Matthew Farrugia-Roberts, Stuart Russell, Alessandro Abate, and Adam Gleave. *Invariance in Policy Optimisation and Partial Identifiability in Reward Learning*. 2022. arXiv: `2203.07475 [cs.LG]`.

[156] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. "Learning to Summarize from Human Feedback". In: *Proceedings of the 34th Conference on Neural Information Processing Systems*. Vol. 33. 2020, pp. 3008–3021.

[157] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018.

[158] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. "Intriguing Properties of Neural Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014.

[159] Aviv Tamar, Shie Mannor, and Huan Xu. "Scaling Up Robust MDPs Using Function Approximation". In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. 2014, pp. 181–189.

[160] Elie Tamer. "Partial Identification in Econometrics". In: *Annual Review of Economics* 2 (2010), pp. 167–195.

[161] Yuandong Tian, Jerry Ma, Qucheng Gong, Shubho Sengupta, Zhuoyuan Chen, James Pinkerton, and Larry Zitnick. "ELF OpenGo: an analysis and open reimplementation of AlphaZero". In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019, pp. 6244–6253.

[162] Finbarr Timbers, Nolan Bard, Edward Lockhart, Marc Lanctot, Martin Schmid, Neil Burch, Julian Schrittwieser, Thomas Hubert, and Michael Bowling. *Approximate exploitability: Learning a best response in large games*. 2022. arXiv: `2004.09677v4 [cs.LG]`.

[163] John Tromp. *The Game of Go*. 2014. URL: `https://tromp.github.io/go.html` (visited on 06/16/2022).

[164] Hsiao-Yu Tung, Adam W Harley, Liang-Kang Huang, and Katerina Fragkiadaki. "Reward Learning from Narrated Demonstrations". In: *Proceedings: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7004–7013.

[165] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aaron van den Oord. "Adversarial Risk and the Dangers of Evaluating Against Weak Attacks". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 5025–5034.

[166] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothörl, Todd Hester, and Jon Scholz. "A Practical Approach to Insertion with Variable Socket Position Using Deep Reinforcement Learning". In: *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)*. 2019.

[167] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. "Programmatically Interpretable Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 5045–5054.

[168] Eugene Vinitsky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. *Robust Reinforcement Learning Using Adversarial Populations*. 2020. arXiv: `2008.01825`.

[169] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[170] Steven Wang, Adam Gleave, and Sam Toyer. *Imitation: Implementations of Inverse Reinforcement Learning and Imitation Learning Algorithms*. 2020. URL: `https://github.com/humancompatibleai/imitation`.

[171] Tony Tong Wang, Adam Gleave, Nora Belrose, Tom Tseng, Joseph Miller, Michael Dennis, Yawen Duan, Viktor Pogrebniak, Sergey Levine, and Stuart Russell. *Adversarial Policies Beat Professional-Level Go AIs*. arXiv: `2211.00241 [cs.LG]`.

[172] Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. "Principled Methods for Advising Reinforcement Learning Agents". In: *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*. 2003, pp. 792–799.

[173] Aaron Wilson, Alan Fern, and Prasad Tadepalli. "A Bayesian Approach for Policy Learning from Trajectory Preference Queries". In: *Advances in Neural Information Processing Systems (NIPS)*. 2012.

[174] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. "A Survey of Preference-Based Reinforcement Learning Methods". In: *Journal of Machine Learning Research* 18.136 (2017), pp. 1–46.

[175] David J. Wu. *Accelerating Self-Play Learning in Go*. 2019. arXiv: `1902.10565 [cs.LG]`.

[176] David J. Wu. *KataGo - Networks for kata1*. 2022. URL: `https://katagotraining.org/networks/` (visited on 09/26/2022).

[177] David J. Wu. *KataGo's Supported Go Rules (Version 2)*. 2021. URL: `https://lightvector.github.io/KataGo/rules.html` (visited on 09/27/2022).

[178] Xian Wu, Wenbo Guo, Hua Wei, and Xinyu Xing. "Adversarial Policy Training against Deep Reinforcement Learning". In: *USENIX Security*. 2021.

[179] Blake Wulfe, Logan Michael Ellis, Jean Mercat, Rowan Thomas McAllister, and Adrien Gaidon. "Dynamics-Aware Comparison of Learned Reward Functions". In: *Proceedings of the Tenth International Conference on Learning Representations*. 2022.

[180] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. "Feature Denoising for Improving Adversarial Robustness". In: *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[181] Brian D Ziebart. "Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy". PhD thesis. Carnegie Mellon University, 2010.

[182] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. "Modeling Interaction via the Principle of Maximum Causal Entropy". In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. 2010, pp. 1255–1262.

[183] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. "Maximum Entropy Inverse Reinforcement Learning". In: *23rd AAAI Conference on Artificial Intelligence*. Vol. 8. 2008, pp. 1433–1438.

[184] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. *Fine-Tuning Language Models from Human Preferences*. 2019. arXiv: `1909.08593 [cs.CL]`.

[185] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. "Regret Minimization in Games with Incomplete Information". In: *NeurIPS*. Vol. 20. 2007.

[186] Xingdong Zuo. *Mazelab: a Customizable Framework to Create Maze and Gridworld Environments*. 2018. URL: `https://github.com/zuoxingdong/mazelab`.

Appendix A

Deferred content from Chapter 3

A.1 Properties of fundamental reward transformations

We begin with some supporting results concerning the basic reward transformations used in Section 3.3 to characterise the invariances of various objects derived from the reward function.

The following result captures how potential shaping affects various reward-related functions.

Lemma A.1.1. *Consider M and M', two MDPs differing only in their reward functions, respectively R and R'. Denote the return function, Q-function, value function, policy evaluation function, and advantage function of M' by G', Q'_π, V'_π, $\mathcal{J}'$, and A'_π. If R' is produced by potential shaping of R with a potential function Φ, then:*

(1) for a trajectory fragment $\zeta = (s_0, a_0, s_1, \ldots, s_n)$, $G'(\zeta) = G(\zeta) + \gamma^n \Phi(s_n) - \Phi(s_0)$;

(2) for a trajectory $\xi = (s_0, a_0, \ldots)$, $G'(\xi) = G(\xi) - \Phi(s_0)$;

(3) for a state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, $Q'_\pi(s, a) = Q_\pi(s, a) - \Phi(s)$;

(4) for a state $s \in \mathcal{S}$, $V'_\pi(s) = V_\pi(s) - \Phi(s)$;

(5) for a policy π, $\mathcal{J}'(\pi) = \mathcal{J}(\pi) - \mathbb{E}_{S_0 \sim \mu_0}[\Phi(S_0)]$; and

(6) for a state $s \in \mathcal{S}$, and action $a \in \mathcal{A}$, $A'_\pi(s, a) = A_\pi(s, a)$.

Proof. (1) is given by a straightforward telescopic argument. For (2), take the limit as the length of a prefix goes to infinity, whereupon $\gamma^n \Phi(s_n)$ goes to zero ($\gamma < 1$ by definition, and $\Phi(s_n)$ is bounded since its domain is finite). (3) and (4) were proved for optimal policies by Ng, Harada, and Russell [111], and they also observed that the extension to arbitrary policies is straightforward (it follows immediately from (2), for example). (5) is immediate from (4). (6) follows from (3) and (4) as the shifts of $-\Phi(s)$ to both the Q- and value functions cancel eachother. □

In the next result we show that potential shaping induces a similar state-dependent shift in the soft Q-function as well.

Lemma A.1.2. *Consider M_1 and M_2, two MDPs differing only in their reward functions, respectively R_1 and R_2. Denote the soft Q-function of M_1 by $Q^{\mathrm{H}}_{\beta,1}$, and of M_2 by $Q^{\mathrm{H}}_{\beta,2}$. If R_2 is produced by potential shaping of R_1 with a potential function Φ, then for all states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$, $Q^{\mathrm{H}}_{\beta,2}(s,a) = Q^{\mathrm{H}}_{\beta,1}(s,a) - \Phi(s)$.*

Proof. We will appeal to to uniqueness of the soft Q-function. By definition, $R_2(s,a,s') = R_1(s,a,s') + \gamma \cdot \Phi(s') - \Phi(s)$. Combining with Equation (3.3), we have for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$:

$$\begin{aligned} Q^{\mathrm{H}}_{\beta,1}(s,a) &= \mathbb{E}_{S' \sim \tau(s,a)}\left[R_1(s,a,S') + \gamma \frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp \beta Q^{\mathrm{H}}_{\beta,1}(S',a')\right] \\ &= \mathbb{E}_{S' \sim \tau(s,a)}\left[R_2(s,a,S') - \gamma \cdot \Phi(S') + \Phi(s) + \gamma \frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp \beta Q^{\mathrm{H}}_{\beta,1}(S',a')\right] \end{aligned}$$

This implies that:

$$Q^{\mathrm{H}}_{\beta,1}(s,a) - \Phi(s) = \mathbb{E}_{S' \sim \tau(s,a)}\left[R_2(s,a,S') + \gamma \frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp \beta \left(Q^{\mathrm{H}}_{\beta,1}(S',a') - \Phi(S')\right)\right].$$

We see that $Q^{\mathrm{H}}_{\beta,1}(s,a) - \Phi(s)$ satisfies Equation (3.3) for $Q^{\mathrm{H}}_{\beta,2}(s,a)$, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Since the soft Q-function is the unique solution to this equation, we conclude $Q^{\mathrm{H}}_{\beta,2}(s,a) = Q^{\mathrm{H}}_{\beta,1}(s,a) - \Phi(s)$. □

We next show that k-initial potential shaping and linear scaling of R correspond to affine transformations of G.

Lemma A.1.3. *Let $(\mathcal{S}, \mathcal{A}, \tau, \mu_0, R, \gamma)$ be an MDP, R' a reward function, and $k \in \mathbb{R}$ a constant. Then we have that $G'(\xi) = G(\xi) - k$ for all possible and initial trajectories ξ, if and only if R' is produced from R by k-initial potential shaping and a mask of unreachable transitions.*

Proof. The converse follows from Lemma A.1.1 and that, by definition, varying the reward for unreachable transitions does not affect the return of any possible, initial trajectories.

For the forward direction, we show that the constant difference between G' and G on possible initial trajectories implies a constant difference between the returns of possible trajectories from any given reachable state, and that this state-dependent difference defines a k-initial potential function that transforms R into R'.

Consider an arbitrary reachable state $s \in \mathcal{S}$. Let ξ_s be some possible trajectory starting in s, and define $\Delta_{\xi_s} = G(\xi_s) - G'(\xi_s)$, the difference in return ascribed to this trajectory by G and G'. We show that Δ_{ξ_s} is independent of ξ_s given s. To extend ξ_s into an initial trajectory, let ζ_s be some possible, initial, trajectory fragment ending in s (at least one exists, since s is reachable; let its length be n). Let $\zeta_s + \xi_s$ denote the concatenation of ζ_s and ξ_s. Then,

$$\begin{aligned}\Delta_{\xi_s} &= G(\xi_s) - G'(\xi_s)\\ &= \frac{G(\zeta_s + \xi_s) - G(\zeta_s)}{\gamma^n} - \frac{G'(\zeta_s + \xi_s) - G'(\zeta_s)}{\gamma^n} &(\dagger)\\ &= \frac{k - G(\zeta_s) + G'(\zeta_s)}{\gamma^n}\,. &(\ddagger)\end{aligned}$$

To reach (†), note that by definition of return, $G(\zeta_s + \xi_s) = G(\zeta_s) + \gamma^n G(\xi_s)$ (and likewise for G'), and recall that we have defined $\gamma > 0$. To reach (‡), note that since $\zeta_s + \xi_s$ is an initial trajectory, we have by assumption $G(\zeta_s + \xi_s) - G'(\zeta_s + \xi_s) = k$. Note (‡) shows that Δ_{ξ_s} is independent of ξ_s except for a possible dependence on ξ_s's starting state s (arising through ζ_s).

Thus, we may associate a unique $P(s) = \Delta_{\xi_s}$ with each reachable s. Then $P(s)$ is a k-initial potential function on reachable states. In particular, $P(s) = \Delta_{\xi_s} = k$ if s is initial as then we may choose ζ_s to be empty with $G(\zeta_s) = G'(\zeta_s) = 0$ and $n = 0$. Furthermore, from the definition of terminal states we must have that $P(s) = \Delta_{\xi_s} = 0$ for terminal s.

Moreover, for reachable transitions, R' is given by k-initial potential shaping of R with $\Phi(s) = P(s)$. Consider a reachable transition (s, a, s'). Let ξ and ξ' be possible trajectories such that $\xi = (s, a, s') + \xi'$. Then,

$$\begin{aligned}R(s,a,s') + \gamma P(s') - P(s) &= R(s,a,s') + \gamma(G(\xi') - G'(\xi')) - (G(\xi) - G'(\xi))\\ &= G'(\xi) - \gamma G'(\xi') + (R(s,a,s') + \gamma G(\xi')) - G(\xi)\\ &= G'(\xi) - \gamma G'(\xi') + G(\xi) - G(\xi)\\ &= G'(\xi) - \gamma G'(\xi')\\ &= R'(s,a,s')\,.\end{aligned}$$

Any variation in reward for unreachable transitions can be accounted for by a mask. □

Lemma A.1.4. *Let $(\mathcal{S}, \mathcal{A}, \tau, \mu_0, R, \gamma)$ be an MDP, R' a reward function, and $c \in \mathbb{R}$ a constant. Then $G'(\xi) = c \cdot G(\xi)$ for all possible initial trajectories ξ, if and only if R' is produced from R by zero-initial potential shaping, linear scaling by a factor of c, and a mask of all unreachable transitions.*

Proof. It is sufficient to show that the first condition is equivalent to R' being produced from $c \cdot R$ by zero-initial potential shaping and a mask of all unreachable transitions (in particular, any sequence of the above three transformations from R can be converted into a sequence where the linear scaling happens first).

Denote by G_c the return function of the scaled reward function $c \cdot R$. It is straightforward to show that $c \cdot G(\xi) = G_c(\xi)$ for all ξ. Then our first condition, $G'(\xi) = c \cdot G(\xi)$ for all possible initial trajectories, is equivalent to having $G'(\xi) = G_c(\xi)$ for these trajectories.

By Lemma A.1.3 (with $k = 0$) this is equivalent to R' being produced from $c \cdot R$ by zero-initial potential shaping and a mask of all unreachable transitions. This completes the proof. □

A.2 Proofs

We provide proofs for the theoretical results presented in the main paper along with several general supporting lemmas from which these results follow.

We distribute proofs of the results from Section 3.3.1 across three subsections. Appendix A.2.1 proves results for the invariances of (soft) Q-functions (Theorems 3.3.1 and 3.3.2). Appendix A.2.2 proves results concerning alternative policies and their trajectory distributions (Theorems 3.3.3 and 3.3.5). Appendix A.2.3 proves the results relating to optimal policies and their trajectory distributions (Theorems 3.3.4 and 3.3.6).

The remaining subsections (Appendices A.2.4 to A.2.6) prove the results from Sections 3.3.2, 3.3.3 and 3.4 respectively.

A.2.1 Proofs for Section 3.3.1 Results Concerning Q-functions

Theorem 3.3.1. *Given an MDP and a policy π, the Q-function for π, Q_π, determines R up to S'-redistribution. The optimal Q-function $Q_\star$ has the same invariances.*

Proof. Q_π is the only function which satisfies the Bellman equation (2.1) for all $s \in \mathcal{S}$, $a \in \mathcal{A}$:

$$Q_\pi(s,a) = \mathbb{E}_{S' \sim \tau(s,a), A' \sim \pi(S')} \left[R(s,a,S') + \gamma \cdot Q_\pi(S', A') \right].$$

This equation can be rewritten as

$$\mathbb{E}_{S' \sim \tau(s,a)} \left[R(s,a,S') \right] = Q_\pi(s,a) - \gamma \cdot \mathbb{E}_{S' \sim \tau(s,a), A' \sim \pi(S')} \left[Q_\pi(S', A') \right].$$

Since Q_π is the only function which satisfies this equation for all $s \in \mathcal{S}, a \in \mathcal{A}$, we have that the values of the left-hand side for each $s \in \mathcal{S}, a \in \mathcal{A}$ together determine Q_π, and vice versa. Since the left-hand side values are preserved by S'-redistribution of R, and no other transformations (cf. Definition 3.2.3), we have that Q_π is preserved by S'-redistribution of R, and no other transformations.

$Q_\star = Q_{\pi_\star}$ where $\pi_\star$ is any optimal policy derived from $Q_\star$, so the invariances of the optimal Q-function follow as a special case.

□

Theorem 3.3.2. *Given an MDP and an inverse temperature parameter β, the soft Q-function Q^{H}_β determines R up to S'-redistribution.*

Proof. The proof is essentially the same as that for Theorem 3.3.1. Q_β^{H} is the only function that satisfies Equation (3.3) for all $s \in \mathcal{S}$, $a \in \mathcal{A}$:

$$Q_\beta^{\mathrm{H}}(s,a) = \mathbb{E}_{S' \sim \tau(s,a)} \left[R(s,a,S') + \gamma \frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp \beta Q_\beta^{\mathrm{H}}(S',a') \right] .$$

This can be rewritten as

$$\mathbb{E}_{S' \sim \tau(s,a)} \left[R(s,a,S') \right] = Q_\beta^{\mathrm{H}}(s,a) - \gamma \cdot \mathbb{E}_{S' \sim \tau(s,a)} \left[\frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp \beta Q_\beta^{\mathrm{H}}(S',a') \right] .$$

Since Q_β^{H} is the only function which satisfies this equation for all $s \in \mathcal{S}, a \in \mathcal{A}$, we have that the values of the left-hand side for each $s \in \mathcal{S}, a \in \mathcal{A}$ together determine Q_β^{H}, and vice versa. Since the left-hand side values are preserved by S'-redistribution of R, and no other transformations (cf. Definition 3.2.3), we have that Q_β^{H} is preserved by S'-redistribution of R, and no other transformations. □

A.2.2 Proofs for Section 3.3.1 Results Concerning Alternative Policies

We split the proof of Theorem 3.3.3 into two proofs, Theorem A.2.3 and Theorem A.2.4, below.

In order to derive the invariances of the Boltzmann-rational policy, we analyse a more general softmax-based policy we call a *Boltzmann policy*, of which the Boltzmann-rational policy is a special case. Given a base policy π_0, and an *inverse temperature* parameter $\beta > 0$, we define the *Boltzmann policy* with respect to π_0, denoted $\pi_\beta^{\pi_0}$, using the softmax function:

$$\pi_\beta^{\pi_0}(a \mid s) = \frac{\exp\big(\beta A_{\pi_0}(s,a)\big)}{\sum_{a' \in \mathcal{A}} \exp\big(\beta A_{\pi_0}(s,a')\big)}. \tag{A.1}$$

The *Boltzmann-rational* policy, $\pi_\beta^\star$, is the Boltzmann policy with respect to optimal policies (cf. 3.1).

We begin with Lemma A.2.1, characterising the invariance of the advantage functions from which Boltzmann policies are derived. This in turn supports Lemma A.2.2, characterising the invariances of arbitrary Boltzmann policies.

Lemma A.2.1. *Given an MDP and a policy π, the advantage function for π, A_π, determines R up to S'-redistribution and potential shaping. The optimal advantage $A_\star$ has the same invariances.*

Proof. A_π can be derived from Q_π, given π (by Equation (2.1), $A_\pi(s,a) = Q_\pi(s,a) - \mathbb{E}_{A \sim \pi(s)} \left[Q_\pi(s,A) \right]$). Thus A_π is invariant to S'-redistribution following Theorem 3.3.1. Moreover, by Lemma A.1.1, potential shaping causes no change in A_π. That is, A_π is also invariant to potential shaping.

Conversely, let R and R' be such that $A_\pi = A'_\pi$. Define $\Phi : \mathcal{S} \to \mathbb{R}$ such that $\Phi(s) = \mathbb{E}_{A \sim \pi(s)}\left[Q_\pi(s, A) - Q'_\pi(s, A)\right]$. This Φ satisfies the requirements of a potential function (all Q-values from terminal states are zero). Potential shaping R with Φ yields a new reward function, denoted $R^{(\Phi)}$, with Q-function denoted $Q_\pi^{(\Phi)}$. Then observe, for each $s \in \mathcal{S}, a \in \mathcal{A}$:

$$\begin{aligned}
Q_\pi^{(\Phi)} &= Q_\pi(s, a) - \Phi(s) && \text{(by Lemma A.1.1)} \\
&= \left(Q_\pi(s, a) - \mathbb{E}_{A \sim \pi(s)}\left[Q_\pi(s, A)\right]\right) + \mathbb{E}_{A \sim \pi(s)}\left[Q'_\pi(s, A)\right] \\
&= A_\pi(s, a) + \mathbb{E}_{A \sim \pi(s)}\left[Q'_\pi(s, A)\right] \\
&= A'_\pi(s, a) + \mathbb{E}_{A \sim \pi(s)}\left[Q'_\pi(s, A)\right] && (A_\pi = A'_\pi \text{ by assumption}) \\
&= \left(Q'_\pi(s, a) - \mathbb{E}_{A \sim \pi(s)}\left[Q'_\pi(s, A)\right]\right) + \mathbb{E}_{A \sim \pi(s)}\left[Q'_\pi(s, A)\right] \\
&= Q'_\pi(s, a)\,.
\end{aligned}$$

That is, $R^{(\Phi)}$ and R' share a Q-function. Thus, by Theorem 3.3.1, R' is given by S'-redistribution from $R^{(\Phi)}$.

The optimal advantage function's invariances arise as a special case, since $A_\star = A_{\pi_\star}$, where $\pi_\star$ is any optimal policy derived from $A_\star$. □

Lemma A.2.2. *Given an MDP, an inverse temperature parameter β, and a base policy π_0, the Boltzmann policy $\pi_\beta^{\pi_0}$ determines R up to S'-redistribution and potential shaping.*

Proof. By Equation (A.1), $\pi_\beta^{\pi_0}$ can be derived from A_{π_0}. Thus $\pi_\beta^{\pi_0}$ is invariant to S'-redistribution and potential shaping by Lemma A.2.1.

Conversely, we show that A_{π_0} can be derived from $\pi_\beta^{\pi_0}$ in turn. Therefore $\pi_\beta^{\pi_0}$ can have no more invariances than A_{π_0}, amounting to S'-redistribution and potential shaping by Lemma A.2.1.

For each $s \in \mathcal{S}, a \in \mathcal{A}$, observe:

$$\begin{aligned}
\pi_\beta^{\pi_0}(a \mid s) &= \frac{\exp\bigl(\beta A_{\pi_0}(s, a)\bigr)}{\sum_{a' \in \mathcal{A}} \exp\bigl(\beta A_{\pi_0}(s, a')\bigr)} \\
\to \quad A_{\pi_0}(s, a) &= \frac{1}{\beta} \log \pi_\beta^{\pi_0}(a \mid s) + \frac{1}{\beta} \log \sum_{a' \in \mathcal{A}} \exp\bigl(\beta A_{\pi_0}(s, a')\bigr)\,. \qquad (\dagger)
\end{aligned}$$

We have not yet solved for A_{π_0}, since it still occurs on both sides of (†). However, we can eliminate the RHS occurrence by appealing to the following identity (that the advantage has zero mean in each state $s \in \mathcal{S}$):

$$\begin{aligned}
\mathbb{E}_{A \sim \pi_0(s)}\left[A_{\pi_0}(s, A)\right] &= \mathbb{E}_{A \sim \pi_0(s)}\left[Q_{\pi_0}(s, A) - \mathbb{E}_{A' \sim \pi_0(s)}\left[Q_{\pi_0}(s, A')\right]\right] \\
&= \mathbb{E}_{A \sim \pi_0(s)}\left[Q_{\pi_0}(s, A)\right] - \mathbb{E}_{A \sim \pi_0(s)}\left[Q_{\pi_0}(s, A)\right] \\
&= 0\,.
\end{aligned}$$

Taking the expectation of both side of (†) therefore yields:

$$\begin{aligned}
\mathbb{E}_{A\sim\pi_0(s)}\left[A_{\pi_0}(s,A)\right] &= \mathbb{E}_{A\sim\pi_0(s)}\left[\frac{1}{\beta}\log\pi_\beta^{\pi_0}(A\mid s)\right] \\
&\quad + \mathbb{E}_{A\sim\pi_0(s)}\left[\frac{1}{\beta}\log\sum_{a'\in\mathcal{A}}\exp\bigl(\beta A_{\pi_0}(s,a')\bigr)\right] \\
\rightarrow\quad 0 &= \mathbb{E}_{A\sim\pi_0(s)}\left[\frac{1}{\beta}\log\pi_\beta^{\pi_0}(A\mid s)\right] + \frac{1}{\beta}\log\sum_{a'\in\mathcal{A}}\exp\bigl(\beta A_{\pi_0}(s,a')\bigr) \\
\rightarrow\quad \frac{1}{\beta}\log\sum_{a'\in\mathcal{A}}\exp\bigl(\beta A_{\pi_0}(s,a')\bigr) &= -\mathbb{E}_{A\sim\pi_0(s)}\left[\frac{1}{\beta}\log\pi_\beta^{\pi_0}(A\mid s)\right]. \qquad (\ddagger)
\end{aligned}$$

Combining (‡) with (†) gives us an expression for A_{π_0} in terms only of $\pi_\beta^{\pi_0}$, as required:

$$A_{\pi_0}(s,a) = \frac{1}{\beta}\log\pi_\beta^{\pi_0}(a\mid s) - \mathbb{E}_{A\sim\pi_0(s)}\left[\frac{1}{\beta}\log\pi_\beta^{\pi_0}(A\mid s)\right].$$

□

Our main result concerning Boltzmann-rational policies follows immediately.

Theorem A.2.3. *Given an MDP and an inverse temperature parameter β, the Boltzmann-rational policy $\pi_\beta^\star$, determines R up to S'-redistribution and potential shaping.*

Proof. The Boltzmann-rational policy $\pi_\beta^\star$ determines its own base policy $\pi_\star$. This is because the maximum probability actions in $\pi_\beta^\star$ are precisely those actions with maximal optimal advantage $A_\star$ ($\arg\max_{a\in\mathcal{A}} A_\star(s) = \arg\max_{a\in\mathcal{A}} \pi_\beta^\star(s,a)$). We can break ties arbitrarily, as any optimal base policy will lead to the same Boltzmann-rational policy. So, given $\pi_\beta^\star$, we are effectively also given a base policy, and the invariances of $\pi_\beta^\star$ therefore follow as a special case of Lemma A.2.2. □

We turn to prove the corresponding result about MCE policies, which follows a similar line of reasoning relative to the soft Q-function. We use an elementary property of the softmax function, which we state and derive as Lemma A.2.5 for the convenience of the unfamiliar reader.

Theorem A.2.4. *Given an MDP and an inverse temperature β, the MCE policy π_β^{H} determines R up to S'-redistribution and potential shaping.*

Proof. π_β^{H} is given by applying the softmax function to Q_β^{H}. Recall (or see Lemma A.2.5, below) that the softmax function is invariant to a constant shift, and no other transformations. This means that π_β^{H} is invariant to exactly those transformations that induce constant shifts in Q_β^{H} for each state.

S'-redistribution induces no shift in Q^{H}_{β} by Theorem 3.3.2. By Lemma A.1.2, potential shaping induces a state-dependent constant shift. Thus, π^{H}_{β} is invariant to S'-redistribution and potential shaping.

Conversely, we show that any state-dependent constant shift in Q^{H}_{β} can be described by these two kinds of transformations. Therefore, they are the only invariances. Let $B : \mathcal{S} \to \mathbb{R}$, and suppose R_1 and R_2 are two reward functions such that the corresponding soft Q-functions satisfy $Q^{\mathrm{H}}_{\beta,1}(s,a) = Q^{\mathrm{H}}_{\beta,2}(s,a) + B(s)$. Then,

$$\begin{aligned}
\mathbb{E}_{S'\sim\tau(s,a)}\left[R_1(s,a,S')\right] &= Q^{\mathrm{H}}_{\beta,1}(s,a) - \mathbb{E}_{S'\sim\tau(s,a)}\left[\gamma\frac{1}{\beta}\log\sum_{a'\in A}\exp\beta Q^{\mathrm{H}}_{\beta,1}(S',a')\right] \\
&= Q^{\mathrm{H}}_{\beta,2}(s,a) + B(s) - \mathbb{E}_{S'\sim\tau(s,a)}\left[\gamma\frac{1}{\beta}\log\sum_{a'\in A}\exp\beta\left(Q^{\mathrm{H}}_{\beta,2}(S',a') + B(S')\right)\right] \\
&= Q^{\mathrm{H}}_{\beta,2}(s,a) + B(s) - \mathbb{E}_{S'\sim\tau(s,a)}\left[\gamma\frac{1}{\beta}\log\left(\sum_{a'\in A}\exp\beta Q^{\mathrm{H}}_{\beta,2}(S',a')\right) + \gamma B(S')\right] \\
&= \mathbb{E}_{S'\sim\tau(s,a)}\left[R_2(s,a,S') + B(s) - \gamma B(S')\right].
\end{aligned}$$

Now set $\Phi(s) = -B(s)$, and we can see that the difference between R and R' is described by potential shaping and S'-redistribution. □

Lemma A.2.5. *Consider two functions $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{X} \to \mathbb{R}$ defined on a finite set $\mathcal{X}$. Then the softmax distributions over f and $f + g$ agree, that is, for all $x \in \mathcal{X}$,*

$$\frac{\exp(f(x) + g(x))}{\sum_{x'\in\mathcal{X}}\exp(f(x') + g(x'))} = \frac{\exp(f(x))}{\sum_{x'\in\mathcal{X}}\exp(f(x'))}, \tag{A.2}$$

if and only if g is a constant function over $\mathcal{X}$.

Proof. This is an elementary property of the softmax function. The forward direction can be seen by manipulating Equation (A.2) as follows:

$$\begin{aligned}
\frac{\exp(f(x) + g(x))}{\exp(f(x))} &= \frac{\sum_{x'\in\mathcal{X}}\exp(f(x') + g(x'))}{\sum_{x'\in\mathcal{X}}\exp(f(x'))} \\
\to \quad g(x) &= \log\left(\frac{\sum_{x'\in\mathcal{X}}\exp(f(x') + g(x'))}{\sum_{x'\in\mathcal{X}}\exp(f(x'))}\right)
\end{aligned}$$

which is constant in x.

The converse can be seen as follows. Assume $g(x) = G$, a constant. Then,

$$\frac{\exp(f(x) + g(x))}{\sum_{x'\in\mathcal{X}}\exp(f(x') + g(x'))} = \frac{\exp(f(x))\cdot\exp(G)}{\left(\sum_{x'\in\mathcal{X}}\exp(f(x'))\right)\cdot\exp(G)} = \frac{\exp(f(x))}{\sum_{x'\in\mathcal{X}}\exp(f(x'))}.$$

□

We continue with results about the trajectories derived from these alternative policies. We once again prove Theorem 3.3.5 in two parts (Theorems A.2.7 and A.2.8). For Boltzmann-rational trajectories, we once again provide a more general lemma concerning arbitrary Boltzmann policies (A.1).

Lemma A.2.6. *Given an MDP M, an inverse temperature β, and a base policy π_0, the distribution of trajectories, $\Delta_\beta^{\pi_0}$, induced by the Boltzmann policy $\pi_\beta^{\pi_0}$ acting in MDP M determines R up to S'-redistribution, potential shaping, and a mask of* unreachable *transitions.*

Proof. That the distribution is invariant to S'-redistribution and potential shaping follows from Lemma A.2.2. The distribution is also invariant to changes in the reward for transitions out of unreachable states, since these rewards cannot affect the policy for reachable states. As a result, the distribution is additionally invariant to a mask of unreachable transitions.

The trajectory distribution can be factored into the separate distributions $\pi_\beta^{\pi_0}(s) \in \Delta(\mathcal{A})$ for each reachable state s by conditioning on a supported prefix trajectory fragment that leads to s and marginalising over subsequent states and actions. Via a similar argument to the proof of Lemma A.2.2, the distribution determines the reward function for transitions (out of these reachable states) up to potential shaping and S'-redistribution (as they affect reachable states). □

Theorem A.2.7. *Given an MDP M and an inverse temperature parameter β, the distribution of trajectories, $\Delta_\beta^\star$, induced by the Boltzmann-rational policy $\pi_\beta^\star$ acting in MDP M, determines R up to S'-redistribution, potential shaping, and a mask of unreachable transitions.*

Proof. As in Theorem 3.3.3, the invariances for the Boltzmann-rational policy's trajectories arises as a special case. □

We turn to prove the corresponding result about MCE policies, which follows a similar line of reasoning as for the Boltzmann trajectories, but relative to the MCE policy instead.

Theorem A.2.8. *Given an MDP M and an inverse temperature parameter β, the distribution of trajectories, $\Delta_\beta^{\mathrm{H}}$, induced by the MCE policy π_β^{H} acting in MDP M determines R up to S'-redistribution, potential shaping, and a mask of unreachable transitions.*

Proof. Directly analogous to the proof of Lemma A.2.6, (relative to Theorem A.2.4). □

A.2.3 Proofs for Section 3.3.1 Results Concerning Optimal Policies

Our results concerning the invariance of optimal policies and their trajectories follow from the following general result connecting optimality preserving transformations to the set of optimal actions in some subset of states.

The key idea of the proof is to establish a link between the value-bounding function Ψ (Definition 3.2.5) and the optimal value function for R' via the Bellman optimality equation. We note that the definition of optimality preserving transformations is designed specifically to elicit this link.

Lemma A.2.9. *Given an MDP M, suppose we have the set of optimal actions for each state in a subset of states $\mathfrak{S} \subseteq \mathcal{S}$. Let $\mathfrak{O}$ be the set of (set-valued) functions $\mathcal{O} : \mathcal{S} \to \mathcal{P}(\mathcal{A})\backslash\{\varnothing\}$ such that $\mathcal{O}(s) = \arg\max_{a \in \mathcal{A}} A_\star(s, a)$ for all $s \in \mathfrak{S}$ (but where $\mathcal{O}$ is unconstrained outside $\mathfrak{S}$). Then, these optimal action sets determine R up to optimality-preserving transformations with $\mathcal{O} \in \mathfrak{O}$.*

Proof. Suppose R' is obtained from M's reward R via an optimality-preserving transformation with some $\mathcal{O} \in \mathfrak{O}$. Let Ψ be the corresponding value-bounding function, that is, a function $\Psi : \mathcal{S} \to \mathbb{R}$ satisfying, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$,

$$\mathbb{E}_{S' \sim \tau(s,a)} \left[R'(s, a, S') + \gamma \cdot \Psi(S')\right] \leqslant \Psi(s) \,, \tag{A.3}$$

with equality if and only if $a \in \mathcal{O}(s)$. Since $\mathcal{O}(s)$ is nonempty (by definition), we have for all $s \in \mathcal{S}$

$$\Psi(s) = \max_{a \in \mathcal{A}} \left(\mathbb{E}_{S' \sim \tau(s,a)} \left[R'(s, a, S') + \gamma \cdot \Psi(S')\right]\right) \,.$$

This recursive condition on Ψ is the Bellman optimality equation for the unique optimal value function, $V'_\star$, of the MDP with transformed reward R'. Therefore, $\Psi(s) = V'_\star(s)$ for all $s \in \mathcal{S}$, and we can rewrite Equation (A.3) as

$$\mathbb{E}_{S' \sim \tau(s,a)} \left[R'(s, a, S') + \gamma \cdot V'_\star(S')\right] \leqslant V'_\star(s) \,, \tag{A.4}$$

with equality only for $a \in \mathcal{O}(s)$.

Now, consider a state $s \in \mathfrak{S}$. By assumption, for this s, $\mathcal{O}(s) = \arg\max_{a \in \mathcal{A}} A_\star(s, a)$. Then for this state, the actions that attain the optimal value bound in Equation (A.4) are these same optimal actions. Therefore, R' induces the same sets of optimal actions from states in $\mathcal{S}$.

Conversely, consider a second MDP M', differing from M only in its reward function, R'. Assume the set of optimal actions in states in $\mathfrak{S}$ agrees with the optimal actions in M for those states. Let $V'_\star$ and $A'_\star$ denote the optimal value and advantage functions for M'. The Bellman optimality equation for M' ensures that, for $s \in \mathcal{S}$,

$$V'_\star(s) = \max_{a \in \mathcal{A}} \left(\mathbb{E}_{S' \sim \tau(s,a)} \left[R'(s, a, S') + \gamma \cdot V'_\star(S')\right]\right) \tag{A.5}$$

with the maximum attained precisely by the actions $a \in \arg\max_{a \in \mathcal{A}}(A'_\star(s, a))$. Setting $\mathcal{O}(s) = \arg\max_{a \in \mathcal{A}}(A'_\star(s, a))$, Equation (A.5) can be rewritten as

$$\mathbb{E}_{S' \sim \tau(s,a)} \left[R'(s, a, S') + \gamma \cdot V_\star(S')\right] \leqslant V_\star(s) \tag{A.6}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, with equality if and only if $a \in \mathcal{O}(s)$.

Now, for $s \in \mathfrak{S}$, we have $\arg\max_{a \in \mathcal{A}}(A'_\star(s, a)) = \arg\max_{a \in \mathcal{A}}(A_\star(s, a))$, because M and M' have matching sets of optimal actions for these states (by assumption). Then, Equation (A.6) shows that R' is produced from R by an optimality-preserving transformation with $\mathcal{O}(s) = \arg\max_{a \in \mathcal{A}}(A'_\star(s, a))$ (and $\Psi(s) = V'_\star(s)$). □

We are now in a position to prove Theorems 3.3.4 and 3.3.6:

Theorem 3.3.4. *Given an MDP, a maximally supportive optimal policy determines R up to optimality-preserving transformations with $\mathcal{O}(s) = \arg\max_a A_\star(s, a)$.*

Proof. By assumption, our optimal policies are maximally supportive. Therefore, their support determine the set of optimal actions from all states. Also by assumption, our maximally supportive optimal policies are determined by the set of optimal actions in each state. Therefore, a maximally supportive optimal policy has the same reward information as the set of optimal policies in each state. Its invariances follow as a special case of Lemma A.2.9, with $\mathfrak{S} = \mathcal{S}$. □

Theorem 3.3.6. *Given an MDP, consider the distribution of trajectories, $\Delta_\star$, induced by a maximally supportive optimal policy. Let $\mathfrak{S}$ be the set of states in supported trajectories. Let $\mathfrak{O}$ be the set of functions $\mathcal{O}$ defined on $\mathcal{S}$ such that $\mathcal{O}(s) = \arg\max_a A_\star(s, a)$ for all $s \in \mathfrak{S}$. $\Delta_\star$ determines R up to optimality-preserving transformations for any $\mathcal{O} \in \mathfrak{O}$.*

Proof. The distribution of trajectories can be factored into separate distributions $\pi_\star(s) \in \Delta(\mathcal{A})$ for each state $s \in \mathfrak{S}$ (in a manner similar to Lemma A.2.6, as proved above). As above, these individual distributions determine and are determined by the set of optimal actions within each of those states. The invariance result therefore follows from Lemma A.2.9. □

Remark A.2.10. As mentioned in Section 3.3, when there are multiple optimal policies, invariances depend on how the given policy is chosen. The proofs above reveal that our assumptions are crucial in connecting maximally supportive optimal policies to optimal action sets. We comment on the motivation for these assumptions and, following our theme of cataloguing partial identifiability, we sketch how the result would change without them.

Assumption (1), that the given policy is maximally supportive, allows us to rule out unsupported actions as suboptimal. Additional reward transformations could become permissible otherwise. As a well-known example, the zero reward function is consistent with any policy if unsupported actions could also be optimal [112]. This more general case is difficult to analyse within our framework, because it is not well-described by transformations or an equivalence relation. The assumption may be demanding, but the consequences of misspecification are mild in the case of policy optimisation – at least the learnt reward function won't allow any suboptimal actions to become optimal.

Assumption (2), that a given policy is computed only from the set of optimal actions in each state, appears to be common. The purpose of this technical assumption is to rule out pathological schemes for encoding additional reward information through the selection of the policy. Through such schemes one could in principle encode the full reward function, for example into the infinite decimal representation of the probability of taking one action over another in some state. Such a selection scheme, even if it was not known to the learner, would remove invariances, as transformations that change the reward function but not the set of optimal states would change the given policy.

A.2.4 Proofs for Section 3.3.2 Results

Theorem 3.3.7. *Given an MDP, the return function restricted to possible trajectory fragments, G_ζ, determines R up to a mask of* impossible *transitions.*

Proof. The result is immediate, since the restricted domain still includes all possible transitions (as length one trajectory fragments with return equal to the reward of the transition), and no fragments with impossible transitions. □

Theorem 3.3.8. *Given an MDP, the return function restricted to possible and initial trajectories, G_ξ, determines R up to zero-initial potential shaping and a mask of* unreachable *transitions.*

Proof. The result follows from Lemma A.1.3 with $k = 0$. □

Theorem 3.3.9. *Given an MDP and an inverse temperature β, the distribution of comparisons of possible trajectory fragments, $\preceq^\zeta_\beta$, determines R up to a mask of* impossible *transitions.*

Proof. Since $\preceq^\zeta_\beta$ can be derived from G_ζ, it is invariant to a mask of impossible transitions by Theorem 3.3.7. Conversely, $\preceq^\zeta_\beta$ determines R for all *possible* transitions. This is because $R(s, a, s')$ is encoded in the Boltzmann distribution of comparisons between the length zero trajectory fragment $\zeta_0 = (s)$ and the length one trajectory fragment $\zeta_1 = (s, a, s')$, and can be recovered as follows:

$$\begin{aligned}
\mathbb{P}(\zeta_0 \preceq^\zeta_\beta \zeta_1) &= \frac{\exp(\beta G(\zeta_1))}{\exp(\beta G(\zeta_0)) + \exp(\beta G(\zeta_1))} \\
&= \frac{\exp(\beta R(s, a, s'))}{\exp(\beta \cdot 0) + \exp(\beta R(s, a, s'))} \\
\rightarrow \qquad R(s, a, s') &= \frac{1}{\beta} \cdot \log\left(\frac{\mathbb{P}(\zeta_0 \preceq^\zeta_\beta \zeta_1)}{1 - \mathbb{P}(\zeta_0 \preceq^\zeta_\beta \zeta_1)}\right).
\end{aligned}$$

Therefore $\preceq^\zeta_\beta$ is invariant to precisely a mask of impossible transitions. □

Theorem 3.3.10. *Given an MDP and an inverse temperature β, the distribution of comparisons of possible and initial trajectories, $\preceq^\xi_\beta$, determines R up to k-initial potential shaping and a mask of* unreachable *transitions.*

Proof. Note that as $\preceq^\xi_\beta$ can be derived from G_ξ, by Theorem 3.3.8, $\preceq^\xi_\beta$ is invariant to zero-initial potential shaping and a mask of unreachable transitions. It is additionally invariant to k-initial potential shaping for arbitrary constants $k \in \mathbb{R}$, and no other transformations: G_ξ can be recovered from $\preceq^\xi_\beta$ up to a constant (we can compare all possible initial trajectories to an arbitrary reference trajectory and recover their relative return using a similar manipulation as above, but we can't determine the return of the reference trajectory). From there, the precise invariance follows from Lemma A.1.3. □

Theorem 3.3.11. *We have the following bounds on the invariances of the noiseless order of possible trajectory fragments, $\preceq_\star^\zeta$. In all MDPs:*

(1) $\preceq_\star^\zeta$ is invariant to positive linear scaling and a mask of impossible transitions; and

(2) $\preceq_\star^\zeta$ is not invariant to transformations other than zero-preserving monotonic transformations or masks of impossible transitions.

Moreover, there exist MDPs attaining each of these bounds.

Proof. For (1), positive linear scaling of reward by a constant c leads to the same scaling of the return of each trajectory fragment, and this always preserves the relation $\preceq_\star^\zeta$, since for any $c > 0$, $c \cdot G(\zeta_1) \leqslant c \cdot G(\zeta_2) \Leftrightarrow G(\zeta_1) \leqslant G(\zeta_2)$ for all pairs of trajectory fragments ζ_1, ζ_2. Moreover, $\preceq_\star^\zeta$ inherits invariance to a mask of impossible transitions from G_ζ (Theorem 3.3.7).

For (2), let R' be produced from R via some transformation that is *neither* a mask of impossible transitions *nor* a zero-preserving monotonic transformation. It must be that either R' fails to preserve the ordinal comparison of two possible transitions, or that it fails to preserve the set of zero-reward possible transitions, compared to R. In the first case, consider two possible transitions whose rewards are not preserved, x_1 and x_2. Without loss of generality, suppose $R(x_1) \leqslant R(x_2)$ but $R'(x_1) > R'(x_2)$. This corresponds to a change in $\preceq_\star^\zeta$'s comparison of the length one trajectories formed from x_1 and x_2, namely $x_1 \preceq_\star^\zeta x_2$ from true to false. Similarly, in the second case, the comparisons between the transition whose reward became or ceased to be zero and a length one trajectory (with return 0) will have changed. Therefore, $\preceq_\star^\zeta$ is not invariant to such transformations.

The bound (1) is attained by the following MDP invariant precisely to positive linear scaling and a mask of impossible transitions. Let $\mathcal{S} = \{s\}$, $\mathcal{A} = \{a_1, a_2\}$, $R(s, a_1, s) = 1$, and $R(s, a_2, s) = 1 + \gamma$. Since $R(s, a_2, s) = R(s, a_1, s) + \gamma R(s, a_1, s)$, the corresponding order relation will contain both $(s, a_2, s) \preceq_\star^\zeta (s, a_1, s, a_1, s)$ and $(s, a_1, s, a_1, s) \preceq_\star^\zeta (s, a_2, s)$. This property requires that $R(s, a_1, s) = (1 + \gamma) \cdot R(s, a_2, s)$, which is preserved only by linear scaling of R. (Non-positive linear scaling is already ruled out by (2)).

The bound (2) is attained by the following MDP invariant to arbitrary zero-preserving monotonic transformations. Let $\mathcal{S} = \{s_1, s_2\}$, $\mathcal{A} = \{a\}$, with possible transitions (s_1, a, s_2) and (s_2, a, s_2), and $R(s_1, a, s_2) > R(s_2, a, s_2) = 0$. Any zero-preserving monotonic transformation of R preserves the ordering of all *possible* trajectory fragments, namely that all nonempty trajectories starting in s_1 have positive return and all other possible trajectories have zero return. □

Theorem 3.3.12. *Given an MDP, the noiseless order of possible and initial trajectories, $\preceq_\star^\xi$, is invariant to (at least) k-initial potential shaping, positive linear scaling, and a mask of unreachable transitions.*

Proof. The pairwise Boltzmann distributions of $\preceq_\beta^\xi$ can be used to derive the noiseless comparisons of $\preceq_\star^\xi$, since the relative return of each pair of trajectories is encoded in each $\mathbb{P}(\xi_1 \preceq_\beta^\zeta \xi_2)$:

$$\xi_1 \preceq_\star^\xi \xi_2 \Leftrightarrow \big(G(\xi_1) \leqslant G(\xi_2)\big) \Leftrightarrow \big(\exp(\beta G(\xi_1)) \leqslant \exp(\beta G(\xi_2))\big) \Leftrightarrow \left(\tfrac{1}{2} \leqslant \mathbb{P}(\xi_1 \preceq_\beta^\zeta \xi_2)\right).$$

Therefore, $\preceq^{\xi}_{\star}$ is invariant to k-initial potential shaping and a mask of unreachable transitions by Theorem 3.3.10.

That $\preceq^{\xi}_{\star}$ is also invariant to positive linear scaling follows from a similar argument as for the first bound in Theorem 3.3.11, proved above. □

Remark A.2.11. Theorem 3.3.12 is a lower bound on the full set of invariances of the noiseless order of possible and initial trajectories. We note the following:

- It is not a tight bound: At least in some MDPs, the order is invariant to additional transformations.

- A slightly tigher lower bound can be achieved by establishing that $\preceq^{\xi}_{\star}$ can be derived from $\preceq^{\zeta}_{\star}$: Consider, for a given trajectory ξ, the sequence of 'prefix' trajectory fragments $\xi^{(0)}, \xi^{(1)}, \xi^{(2)}, \ldots$, with each $\xi^{(n)}$ comprising the first n transitions of ξ. By definition $G(\xi) = \lim_{n\to\infty} G(\xi^{(n)})$, and so for each pair of trajectories ξ_1, ξ_2, we have $\xi_1 \preceq^{\xi}_{\star} \xi_2$ if and only if $\xi_1^{(n)} \preceq^{\zeta}_{\star} \xi_2^{(n)}$ for infinitely many n. While this is not a practical method to compute the trajectory order $\preceq^{\xi}_{\star}$ from the fragment order $\preceq^{\zeta}_{\star}$, it counts as a derivation in that it is sufficient to show that if a transformation does not change the fragment order $\preceq^{\zeta}_{\star}$, it cannot change the trajectory order $\preceq^{\xi}_{\star}$ either. Therefore, in particular, $\preceq^{\xi}_{\star}$ inherits invariance to ZPMTs in *some* MDPs from $\preceq^{\zeta}_{\star}$. This tightens the bound, at least in some MDPs.

- The previous point does not imply that the trajectory order $\preceq^{\xi}_{\star}$ inherits the fragment order $\preceq^{\zeta}_{\star}$'s *non*-invariances. A case in point is that $\preceq^{\xi}_{\star}$ is invariant to k-initial potential shaping and a mask of unreachable transitions, where $\preceq^{\zeta}_{\star}$ is not (Theorem 3.3.11). It is not yet clear if there are MDPs where $\preceq^{\xi}_{\star}$ is invariant to no ZPMTs other than positive linear scaling, or even to not all ZPMTs, and there may be invariances of $\preceq^{\xi}_{\star}$ that require new transformation classes to describe. However, Theorem 3.3.12 and this remark give us enough information to confidently position $\preceq^{\xi}_{\star}$ in our partial order of reward-derived objects (see Figure 3.1).

Theorem 3.3.13. *Given an MDP, $\preceq^{\xi}_{\mathcal{D}}$ determines R up to k-initial potential shaping, positive linear scaling, and a mask of unreachable transitions.*

Proof. It is clear that preferences between lotteries over a choice set are preserved by positive affine transformations of the value (and no other transformations). In particular, the converse is a consequence of the well-known VNM utility theorem [110]. The proof by Neumann and Morgenstern [110] covers a finite number of outcomes, and the result also holds for an infinite number of outcomes [see, e.g., 48].

Thus, our result is immediate from Lemmas A.1.3 and A.1.4, which together state that these positive affine transformations of the return function correspond exactly to k-initial potential shaping, positive linear scaling, and a mask of unreachable transitions. □

A.2.5 Proofs for Section 3.3.3 Results

Theorem 3.3.14. *Given an MDP, the set of optimal policies determines R up to optimality-preserving transformations with $\mathcal{O}(s) = \arg\max_a A_\star(s,a)$.*

Proof. The set of all optimal policies determines a maximally supportive policy, for example by constructing a policy that supports all actions supported by any policy in the set. Likewise, a maximally supportive policy determines the set of optimal policies, namely as the set of all policies whose support is a subset of the maximally supportive policy. Therefore, these objects share precisely the same invariances. □

A.2.6 Proofs for Section 3.4 Results

Theorem 3.4.2. *Consider an MDP $(\mathcal{S}, \mathcal{A}, \tau, \mu_0, R_1, \gamma)$ and alternative transition dynamics τ'. Given any function $\mathcal{L} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, there exists a reward function R_2, produced from R_1 by S'-redistribution under τ, such that $\mathbb{E}_{S' \sim \tau'(s,a)}\left[R_2(s,a,S')\right] = \mathcal{L}(s,a)$ for all s, a such that $\tau(s,a) \neq \tau'(s,a)$.*

Proof. Per Definition 3.2.3, that R_2 is produced from R_1 by S'-redistribution under τ requires that, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$,

$$\mathbb{E}_{S' \sim \tau(s,a)}\left[R_1(s,a,S')\right] = \mathbb{E}_{S' \sim \tau(s,a)}\left[R_2(s,a,S')\right] \ . \tag{A.7}$$

Let $s \in \mathcal{S}$ and $a \in \mathcal{A}$ be any state and action such that $\tau'(s,a) \neq \tau(s,a)$. Let $\vec{\tau}_{s,a}$ and $\vec{\tau'}_{s,a}$ be $\tau(s,a)$ and $\tau'(s,a)$ expressed as vectors, and let $\vec{R_1}_{s,a}$ be the vector where $\vec{R_1}_{s,a}^{(i)} = R_1(s,a,s_i)$. The question is then if there is an analogous vector $\vec{R_2}_{s,a}$ such that:

$$\begin{aligned} \vec{\tau}_{s,a} \cdot \vec{R_2}_{s,a} &= \vec{\tau}_{s,a} \cdot \vec{R_1}_{s,a} \ , \\ \vec{\tau'}_{s,a} \cdot \vec{R_2}_{s,a} &= \mathcal{L}(s,a). \end{aligned} \tag{A.8}$$

Since $\vec{\tau}_{s,a}$ and $\vec{\tau'}_{s,a}$ differ and are valid probability distributions, they are linearly independent. Therefore, the system of equations (A.8) always has a solution for $\vec{R_2}_{s,a}$. Form the required R_2 as R_1 modified to have the values of $\vec{R_2}_{s,a}$ in these states where the transition function is disturbed. □

Theorem 3.4.3. *Given data sources X and Y, let (X,Y) denote the combined data source formed from X and Y. If X and Y are incomparable, then $(X,Y) \prec X$ and $(X,Y) \prec Y$.*

Proof. Transformations that preserve (X,Y) necessarily preserve X, therefore $(X,Y) \preceq X$. But since X and Y are incomparable, there is some transformation that preserves X and not Y. This transformation does not preserve (X,Y). Therefore, $(X,Y) \prec X$. Similarly, $(X,Y) \prec Y$. □

We note that the above result is also an elementary consequence of the *lattice structure* of the partial order of partition refinement [2, §I.2.B], since the combined data source corresponds to the *meet* of the original data sources.

A.3 Other spaces of reward functions

Hitherto, we have assumed reward functions are members of $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. That is, they are deterministic functions of transitions depending on the state, action, and the successor state. In this appendix, we discuss several alternative spaces of reward functions and their implications for the invariance properties of various objects derived from the reward function.

A.3.1 Restricted-domain Reward Functions

It is common in both reinforcement learning and reward learning to consider less expressive spaces of reward functions. In particular, the domain of the reward function is often restricted to $\mathcal{S}$ or $\mathcal{S} \times \mathcal{A}$. When modelling a task, the choice of reward function domain is usually a formality: An MDP taking full advantage of the domain $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ has an "equivalent" MDP with a restricted domain and some added auxiliary states [139, §17]. Conversely, reward functions with restricted domains can be viewed as a special case of functions from $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ where the functions are constant in the final argument(s). Restricting the domain can be an appealing simplification when modelling a task, hence the popularity of these formulations.

When modelling a data source, this equivalence may not apply: We may not have access to data regarding auxiliary states, so assuming a restricted domain effectively assumes the latent reward is indeed constant with respect to the successor state (and possibly the action) of each transition. This assumption may or may not be warranted.

If a restricted domain of $\mathcal{S}$ or $\mathcal{S} \times \mathcal{A}$ is preferred, then our invariance results can be adapted in a straightforward manner. In general, since we are effectively considering a subspace of candidate reward functions for transformations, ambiguity can only decrease. In particular, these restrictions have two main consequences.

Firstly, the reward function transformation of S'-redistribution vanishes to the identity transformation, since it allows variation only in the successor state argument of the reward function, which is now impossible. This reduces the effective ambiguity of the Q-function and all derivative objects. Notably, the Q-function uniquely identifies the reward function, and Boltzmann policies have the same invariances as Boltzmann comparisons between trajectories. Restricting the domain to $\mathcal{S}$ means the (state) value function for an arbitrary known policy also uniquely identifies the reward function but doesn't otherwise alter the invariances we have explored.

Secondly, for most MDPs, the available potential-shaping transformations are restricted, but not eliminated. The function added in a potential-shaping transformation $(\gamma \cdot \Phi(s') - \Phi(s))$ nominally depends on the successor state of the transition. Some transformed reward functions may rely on this dependence, falling outside of the restricted domain. However, some non-zero transformations will usually remain. For example, in a discounted MDP without terminal states, a non-zero constant potential function $\Phi(s) = k$ does not *effectively* depend on s, and the reward transformation of adding $\gamma \cdot \Phi(s') - \Phi(s) = (\gamma - 1) \cdot k$ to a reward function does not introduce a dependence on s'. In general, the set of remaining potential-shaping transformations will depend on the network structure of the MDP. At the extreme, in a deterministic MDP with state-action rewards, all potential-shaping transformations are permitted, since a dependence on s' can be satisfied by a.

A.3.2 Stochastic Reward Functions

Certain tasks are naturally modelled as providing rewards drawn stochastically from some distribution upon each transition. An even more expressive space of reward functions than we consider is the space of *transition-conditional reward distributions.*[*] Identifying the reward function in this case is more challenging in general because the latent parameter contains a full distribution of information for each input, rather than a single point. In the spirit of this paper, we sketch a characterisation of this additional ambiguity.

A deterministic reward function can be viewed as the conditional expectation of a reward distribution function. Taking the expectation of the reward distribution for each transition introduces invariance, since the expectation operation is not injective (except in certain restricted cases such as for parametric families of distributions that can be parametrised by their mean). The invariance introduced is akin to S'-redistribution, but with an expectation over the support of the reward distribution rather than the successor state of each transition.

In the extension of the RL formalism to account for stochastic rewards, this expectation is effectively the first step in the derivation of each of the objects we have studied. Therefore, all of these objects inherit this new invariance.

As a consequence, all data sources are effectively more ambiguous with respect to this new latent parameter. For example, if optimal comparisons between trajectories are understood to be performed based on the pairwise comparison of the expected return of each individual trajectory, then these comparisons are also invariant to transformations of the reward distributions that preserve their means.

Fortunately, much of reinforcement learning also focuses on expected return and reward *in application*. Accordingly, most downstream tasks are tolerant to any ambiguity in the exact distribution of stochastic rewards, beyond identifying the mean. Since this is the same kind of ambiguity that is introduced by considering the latent parameter of reward learning as a conditional distribution rather than a deterministic function, our results are still informative for these situations.

A.3.3 Further Spaces and Future Work

For certain applications, including *risk-sensitive RL* where non-mean objectives are pursued [107, 108, 38], the distribution of stochastic rewards can be consequential. Moreover, the introduction of stochastic rewards suggests considering data sources based on samples rather than expectations, such as a data source of trajectory comparisons based on sampled trajectory returns. Characterising the invariances of these objectives to transformations of the reward distribution, and thereby their *ambiguity tolerance*, is left to future work.

[*] Of course, it's also possible to consider reward to be distributed conditionally on only the state or state-action components of a transition and not the full transition.

In future extensions of this work to handle continuous MDPs, there will be an opportunity to study the effect of restricting to various parametrised spaces of reward functions. For example, it is common in reinforcement learning and reward learning to study MDPs with reward functions that are linear in a *feature vector* associated with each transition. This kind of restriction may reduce the available reward transformations compared to those available to a non-parametric reward function in a similar manner to restricting the domain of a finite reward function as discussed above.

The relaxation of the Markovian assumption also introduces a broader space of reward functions and with it new dimensions for transformations and invariance. As one example related to potential shaping, the non-Markovian additive transformations studied by Wiewiora, Cottrell, and Elkan [172] will amount to new invariances of the optimal policy and other related objects.

Appendix B

Deferred content from Chapter 4

B.1 Approximation procedures

B.1.1 Sample-based approximation for EPIC distance

We approximate EPIC distance (definition 4.2.7) by estimating Pearson distance on a set of samples, canonicalizing the reward on-demand. Specifically, we sample a batch B_V of N_V samples from the coverage distribution $\mathcal{D}$, and a batch B_M of N_M samples from the joint state and action distributions $\mathcal{D}_\mathcal{S} \times \mathcal{D}_\mathcal{A}$. For each $(s, a, s') \in B_V$, we approximate the canonically shaped rewards (definition 4.2.2) by taking the mean over B_M:

$$\begin{aligned} C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R\right)\left(s, a, s'\right) &= R(s, a, s') + \mathbb{E}\left[\gamma R(s', A, S') - R(s, A, S') - \gamma R(S, A, S')\right] && \text{(B.1)} \\ &\approx R(s, a, s') + \frac{\gamma}{N_M} \sum_{(x,u)\in B_M} R(s', u, x) && \text{(B.2)} \\ &\quad - \frac{1}{N_M} \sum_{(x,u)\in B_M} R(s, u, x) - c. && \text{(B.3)} \end{aligned}$$

We drop the constant c from the approximation since it does not affect the Pearson distance; it can also be estimated in $O(N_M^2)$ time by
$c = \frac{\gamma}{N_M^2} \sum_{(x,\cdot)\in B_M} \sum_{(x',u)\in B_M} R(x, u, x')$. Finally, we compute the Pearson distance between the approximate canonically shaped rewards on the batch of samples B_V, yielding an $O(N_V N_M)$ time algorithm.

B.1.2 Optimization-based approximation for NPEC distance

$D_{\text{NPEC}}(R_A, R_B)$ (section 4.3.2) is defined as the infimum of L^p distance over an infinite set of equivalent reward functions $R \equiv R_A$. We approximate this using gradient descent on the reward model

$$R_{\nu,c,w}(s, a, s') = \exp(\nu) R_A(s, a, s') + c + \gamma \Phi_w(s') - \Phi_w(s), \tag{B.4}$$

where $\nu, c \in \mathbb{R}$ are scalar weights and w is a vector of weights parameterizing a deep neural network Φ_w. The constant $c \in \mathbb{R}$ is unnecessary if Φ_w has a bias term, but its inclusion simplifies the optimization problem.

We optimize ν, c, w to minimize the mean of the cost

$$J(\nu, c, w)(s, a, s') = \|R_{\nu,c,w}(s, a, s'), R_B(s, a, s')\|^p \tag{B.5}$$

on samples (s, a, s') from a coverage distribution $\mathcal{D}$. Note

$$\mathbb{E}_{(S,A,S')\sim\mathcal{D}} \left[J(\nu, c, w)(S, A, S')\right]^{1/p} = D_{L^p,\mathcal{D}}(R_{\nu,c,w}, R_B) \tag{B.6}$$

upper bounds the true NPEC distance since $R_{\nu,c,w} \equiv R_A$.

We found empirically that ν and c need to be initialized close to their optimal values for gradient descent to reliably converge. To resolve this problem, we initialize the affine parameters to $\nu \leftarrow \log \lambda$ and c found by:

$$\underset{\lambda \geqslant 0, c \in \mathbb{R}}{\arg\min} \underset{s,a,s' \sim \mathcal{D}}{\mathbb{E}} \left(\lambda R_A(s, a, s') + c - R_B(s, a, s')\right)^2 . \tag{B.7}$$

We use the active set method of Lawson and Hanson [87] to solve this constrained least-squares problem. These initial affine parameters minimize the L^p distance

$$D_{L^p,\mathcal{D}}(R_{\nu,c,0}(s, a, s'), R_B(s, a, s'))$$

when $p = 2$ with the potential fixed at $\Phi_0(s) = 0$.

B.1.3 Confidence Intervals

We report confidence intervals to help measure the degree of error introduced by the approximations. Since approximate distances may not be normally distributed, we use bootstrapping to produce a distribution-free confidence interval. For EPIC, NPEC and Episode Return (sometimes reported as regret rather than return), we compute independent approximate distances or returns over different seeds, and then compute a bootstrapped confidence interval for each seed. We use 30 seeds for EPIC, but only 9 seeds for computing Episode Return and 3 seeds for NPEC due to their greater computational requirements. In ERC, computing the distance is very fast, so we instead apply bootstrapping to the collected *episodes*, computing the ERC distance for each bootstrapped episode sample.

B.2 Experiments

B.2.1 Hyperparameters for Approximate Distances

Table B.1 summarizes the hyperparameters and distributions used to compute the distances between reward functions. Most parameters are the same across all environments. We

Table B.1: Summary of hyperparameters and distributions used in experiments. The uniform random coverage distribution $\mathcal{D}_{\text{unif}}$ samples states and actions uniformly at random, and samples the next state from the transition dynamics. Random policy π_{uni} takes uniform random actions. The synthetic expert policy π^* was trained with PPO on the ground-truth reward. **Mix**ture samples actions from either π_{uni} or π^*, switching between them at each time step with probability 0.05. Warmstart Size is the size of the dataset used to compute initialization parameters described in section B.1.2. See Table B.2 for the policy training hyperparameters.

Parameter	**Value**	**In experiment**
Coverage Distribution $\mathcal{D}$	Random transitions $\mathcal{D}_{\text{unif}}$	`GridWorld`

	Rollouts from π_{uni}	`PointMass`, `HalfCheetah`
		`Hopper`

	π_{uni}, π^* and **Mix**ture	`PointMaze`
Bootstrap Samples	10 000	All
Discount γ	0.99	All
EPIC		
State Distribution $\mathcal{D}_{\mathcal{S}}$	Marginalized from $\mathcal{D}$	All
Action Distribution $\mathcal{D}_{\mathcal{A}}$	Marginalized from $\mathcal{D}$	All
Seeds	30	All
Samples N_V	32 768	All
Mean Samples N_M	32 768	All
NPEC		
Seeds	3	All
Total Time Steps	1×10^6	All
Optimizer	Adam	All
Learning Rate	1×10^{-2}	All
Batch Size	4096	All
Warmstart Size	16 386	All
Loss ℓ	$\ell(x, y) = (x - y)^2$	All
ERC		
Episodes	131 072	All
Episode Return		
Seeds	9	All

use a coverage distribution of uniform random transitions $\mathcal{D}_{\text{unif}}$ in the simple `GridWorld` environment with known deterministic dynamics. In other environments, the coverage distribution is sampled from rollouts of a policy. We use a random policy π_{uni} for `PointMass`, `HalfCheetah` and `Hopper` in the hand-designed reward experiments (section 4.4.1). In `PointMaze`, we compare three coverage distributions (section 4.4.2) induced by rollouts of π_{uni}, an expert policy π^* and a **Mix**ture of the two policies, sampling actions from either π_{uni} or π^* and switching between them with probability 0.05 per time step.

B.2.2 Training Learned Reward Models

For the experiments on learned reward functions (sections 5.2 and 4.4.2), we trained reward models using adversarial inverse reinforcement learning (AIRL) [49]), preference comparison [33] and by regression onto the ground-truth reward [*target* method from 33, section 3.3]. For AIRL, we use an existing open-source implementation [170]. We developed new implementations for preference comparison and regression, available at `https://github.com/HumanCompatibleAI/evaluating-rewards`. We also use the RL algorithm proximal policy optimization (PPO) [144]) on the ground-truth reward to train expert policies to provide demonstrations for AIRL. We use 9 seeds, taking rollouts from the seed with the highest ground-truth return.

Our hyperparameters for PPO in Table B.2 were based on the defaults in Stable Baselines [65]. We only modified the batch size and learning rate, and disabled value function clipping to match the original PPO implementation.

Our AIRL hyperparameters in Table B.3 likewise match the defaults, except for increasing the total number of timesteps to 10^6. Due to the high variance of AIRL, we trained 5 seeds, selecting the one with the highest ground-truth return. While this does introduce a positive bias for our AIRL results, in spite of this AIRL performed worse in our tests than other algorithms. Moreover, the goal in this paper is to evaluate distance metrics, not reward learning algorithms.

For preference comparison we performed a sweep over batch size, trajectory length and learning rate to decide on the hyperparameters in Table B.4. Total time steps was selected once diminishing returns were observed in loss curves. The exact value of the regularization weight was found to be unimportant, largely controlling the scale of the output at convergence.

Finally, for regression we performed a sweep over batch size, learning rate and total time steps to decide on the hyperparameters in Table B.5. We found batch size and learning rate to be relatively unimportant with many combinations performing well, but regression was found to converge slowly but steadily requiring a relatively large 10×10^6 time steps for good performance in our environments.

All algorithms are trained on synthetic data generated from the ground-truth reward function. AIRL is provided with a large demonstration dataset of 100 000 time steps from an expert policy trained on the ground-truth reward (see Table B.3). In preference comparison and regression, each batch is sampled afresh from the coverage distribution specified in Table B.1 and labeled according to the ground-truth reward.

Table B.2: Hyperparameters for proximal policy optimization (PPO) [144]. We used the implementation and default hyperparameters from Hill et al. [65]. PPO was used to train expert policies on ground-truth reward and to optimize learned reward functions for evaluation.

Parameter	**Value**	**In environment**
Total Time Steps	1×10^6	All
Seeds	9	All
Batch Size	4096	All
Discount γ	0.99	All
Entropy Coefficient	0.01	All
Learning Rate	3×10^{-4}	All
Value Function Coefficient	0.5	All
Gradient Clipping Threshold	0.5	All
Ratio Clipping Thrsehold	0.2	All
Lambda (GAE)	0.95	All
Minibatches	4	All
Optimization Epochs	4	All
Parallel Environments	8	All

Table B.3: Hyperparameters for adversarial inverse reinforcement learning (AIRL) used in Wang, Gleave, and Toyer [170].

Parameter	**Value**
RL Algorithm	PPO [144]
Total Time Steps	1 000 000
Discount γ	0.99
Demonstration Time Steps	100 000
Generator Batch Size	2048
Discriminator Batch Size	50
Entropy Weight	1.0
Reward Function Architecture	MLP, two 32-unit hidden layers
Potential Function Architecture	MLP, two 32-unit hidden layers

Table B.4: Hyperparameters for preference comparison used in our implementation of Christiano et al. [33].

Parameter	**Value**	**Range Tested**
Total Time Steps	5×10^6	$[1, 1 \times 10^7]$
Batch Size	10 000	$[500, 250\,000]$
Trajectory Length	5	$[1, 100]$
Learning Rate	10^{-2}	$[10^{-4}, 10^{-1}]$
Discount γ	0.99	
Reward Function Architecture	MLP, 2 32-unit hidden layers	
Output L2 Regularization Weight	10^{-4}	

Table B.5: Hyperparameters for regression used in our implementation of Christiano et al. [33, *target* method from section 3.3].

Parameter	**Value**	**Range Tested**
Total Time Steps	10×10^6	$[1, 20 \times 10^6]$
Batch Size	4096	$[256, 16\,384]$
Learning Rate	2×10^{-2}	$[10^{-3}, 10^{-1}]$
Discount γ	0.99	
Reward Function Architecture	MLP, two 32-unit hidden layers	

B.2.3 Computing infrastructure

Experiments were conducted on a workstation (Intel i9-7920X CPU with 64 GB of RAM), and a small number of `r5.24xlarge` AWS VM instances, with 48 CPU cores on an Intel Skylake processor and 768 GB of RAM. It takes less than three weeks of compute on a single `r5.24xlarge` instance to run all the experiments described in this paper.

B.2.4 Comparing hand-designed reward functions

We compute distances between hand-designed reward functions in four environments: `GridWorld`, `PointMass`, `HalfCheetah` and `Hopper`. The reward functions for `GridWorld` are described in Figure B.1, and the distances are reported in Figure B.2. We report the approximate distances and confidence intervals between reward functions in the other environments in Figures B.3, B.4 and B.5.

We find the (approximate) EPIC distance closely matches our intuitions for similarity between the reward functions. NPEC often produces similar results to EPIC, but unfortunately is dogged by optimization error. This is particularly notable in higher-dimensional environments like `HalfCheetah` and `Hopper`, where the NPEC distance often exceeds the theoretical upper bound of 1.0 and the confidence interval width is frequently larger than 0.2.

By contrast, ERC distance generally has a tight confidence interval, but systematically fails in the presence of shaping. For example, it confidently assigns large distances between *equivalent* reward pairs in `PointMass` such as `S`🐆-`D`🐆. However, ERC produces reasonable results in `HalfCheetah` and `Hopper` where rewards are all similarly shaped. In fact, ERC picks up on a detail in `Hopper` that EPIC misses: whereas EPIC assigns a distance of around 0.71 between all rewards of different types (running vs backflipping), ERC assigns lower distances when the rewards are in the same direction (forward or backward). Given this, ERC may be attractive in some circumstances, especially given the ease of implementation. However, we would caution against using it in isolation due to the likelihood of misleading results in the presence of shaping.

B.2.5 Comparing learned reward functions

Previously, we reported the mean approximate distance from a ground-truth reward of four learned reward models in `PointMaze` (Table 4.2). Since these distances are approximate, we report 95% lower and upper bounds computed via bootstrapping in Table B.7. We also include the relative difference of the upper and lower bounds from the mean, finding the relative difference to be fairly consistent across reward models for a given algorithm and coverage distribution pair. The relative difference is less than 1% for all EPIC and ERC distances. However, NPEC confidence intervals can be as wide as 50%: this is due to the method's high variance, and the small number of seeds we were able to run because of the method's computational expense.

B.2.6 Runtime of Distance Metrics

We report the empirical runtime for EPIC and baselines in Table B.6, performing 25 pairwise comparisons across 5 reward functions in `PointMass`. These comparisons were run on an unloaded machine running Ubuntu 20.04 (kernel 5.4.0-52) with an Intel i9-7920X CPU and 64 GB of RAM. We report sequential runtimes: runtimes for all methods could be decreased further by parallelizing across seeds. The algorithms were configured to use 8 parallel environments for sampling. Inference and training took place on CPU. All methods used the same TensorFlow configuration, parallelizing operations across threads both within and between operations. We found GPUs offered no performance benefit in this setting, and in some cases even increased runtime. This is due to the fixed cost of CPU-GPU communication, and the relatively small size of the observations.

We find that in just 17 seconds EPIC can provide results with a 95% confidence interval < 0.023, an order of magnitude tighter than NPEC running for over 8 hours. Training policies for all learned rewards in this environment using PPO is comparatively slow, taking over 4 hours even with only 3 seeds. While ERC is relatively fast, it takes a large number of samples to achieve tight confidence intervals. Moreover, since `PointMass` has stochastic initial states, ERC can take on arbitrary values under shaping, as discussed in sections 4.3.1 and 5.2.

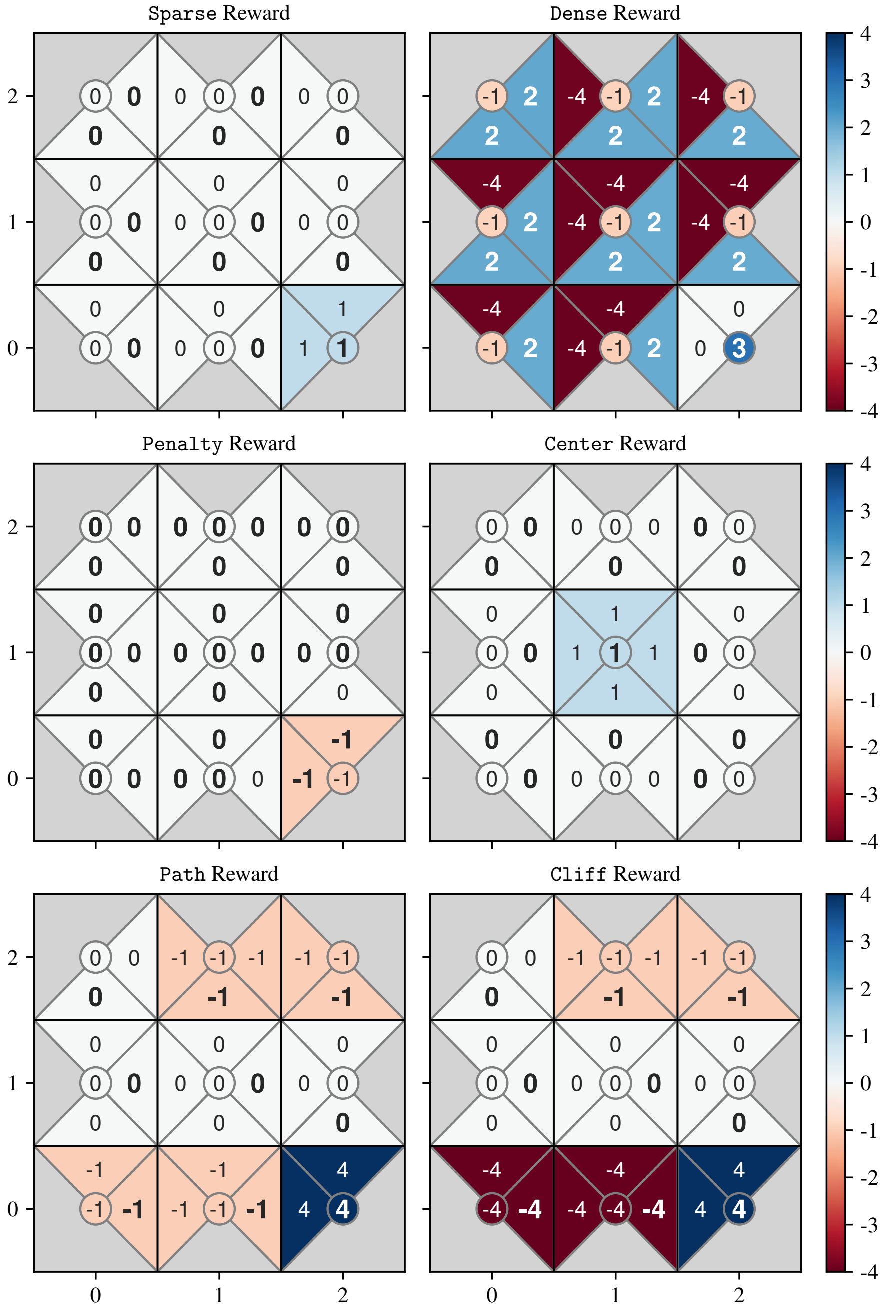
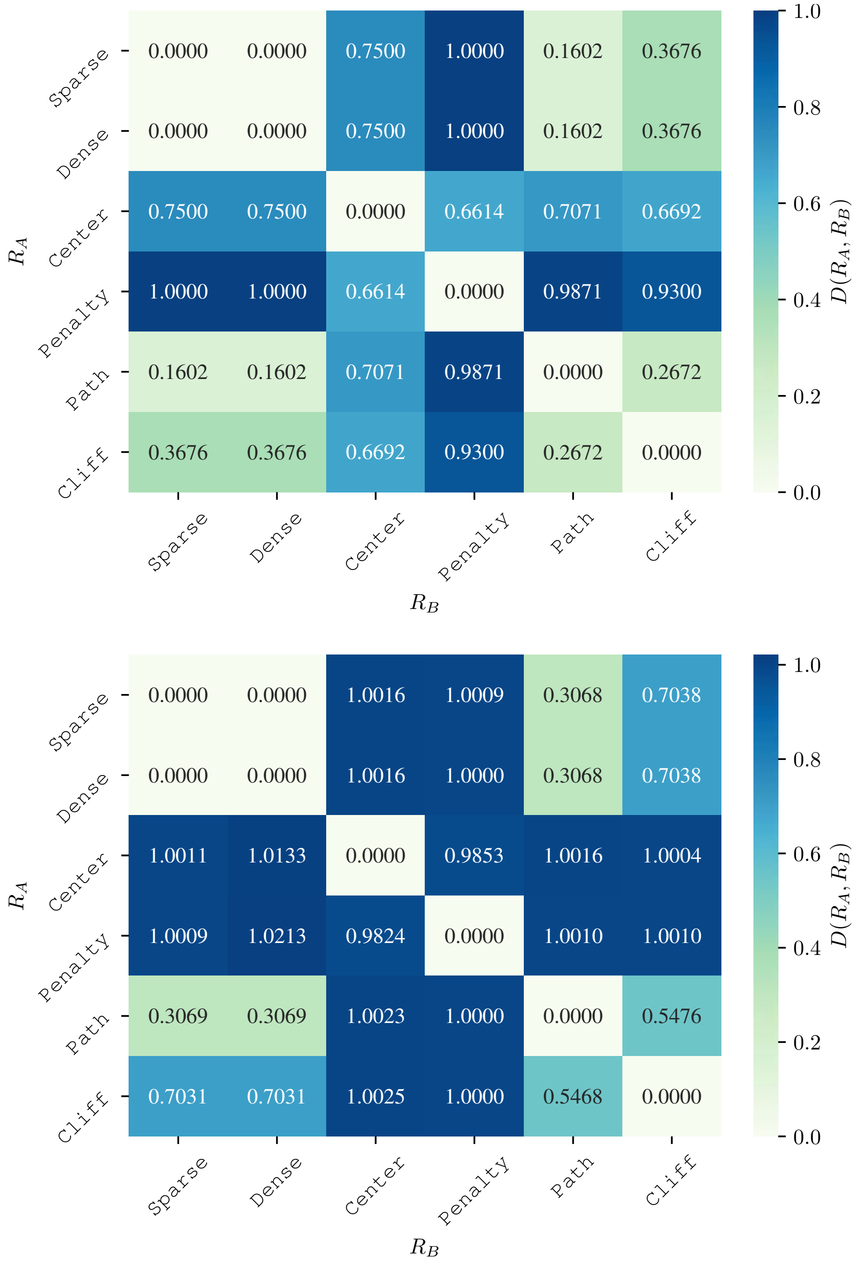


Figure B.1: Heatmaps of reward functions $R(s, a, s')$ for a 3×3 deterministic gridworld. $R(s, \text{stay}, s)$ is given by the central circle in cell s. $R(s, a, s')$ is given by the triangular wedge in cell s adjacent to cell s' in direction a. Optimal action(s) (for infinite horizon, discount $\gamma = 0.99$) have bold labels against a hatched background. See Figure B.2 for the distance between all reward pairs.



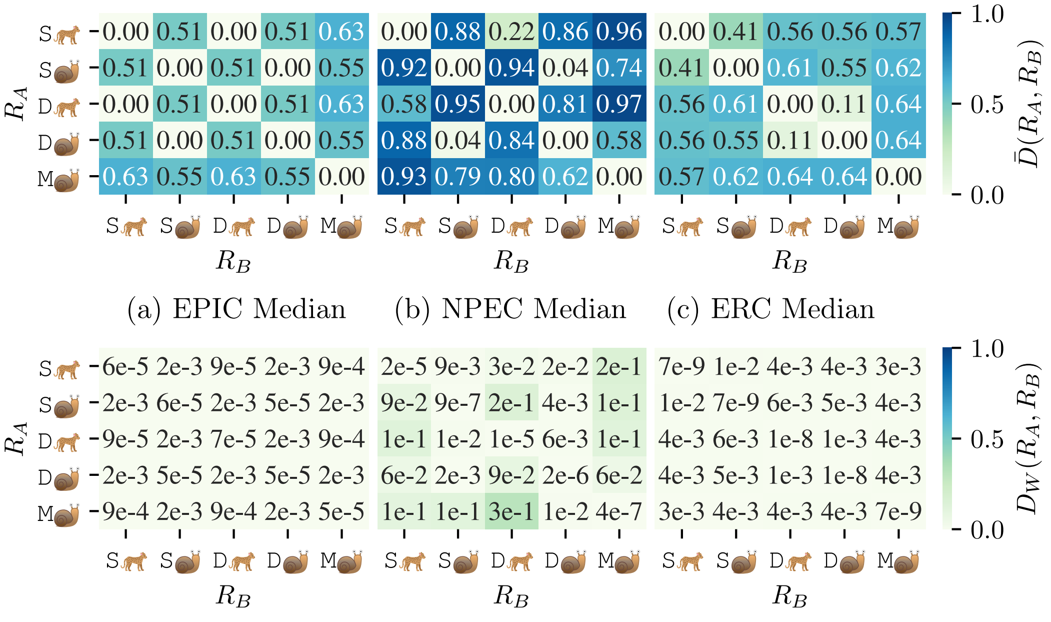
R_A \ R_B	Sparse	Dense	Center	Penalty	Path	Cliff
Sparse	0.0000	0.0000	0.7500	1.0000	0.1602	0.3676
Dense	0.0000	0.0000	0.7500	1.0000	0.1602	0.3676
Center	0.7500	0.7500	0.0000	0.6614	0.7071	0.6692
Penalty	1.0000	1.0000	0.6614	0.0000	0.9871	0.9300
Path	0.1602	0.1602	0.7071	0.9871	0.0000	0.2672
Cliff	0.3676	0.3676	0.6692	0.9300	0.2672	0.0000

R_A \ R_B	Sparse	Dense	Center	Penalty	Path	Cliff
Sparse	0.0000	0.0000	1.0016	1.0009	0.3068	0.7038
Dense	0.0000	0.0000	1.0016	1.0000	0.3068	0.7038
Center	1.0011	1.0133	0.0000	0.9853	1.0016	1.0004
Penalty	1.0009	1.0213	0.9824	0.0000	1.0010	1.0010
Path	0.3069	0.3069	1.0023	1.0000	0.0000	0.5476
Cliff	0.7031	0.7031	1.0025	1.0000	0.5468	0.0000

Figure B.2: Distances (EPIC, top; NPEC, bottom) between hand-designed reward functions for the 3×3 deterministic `Gridworld` environment. EPIC and NPEC produce similar results, but EPIC more clearly discriminates between rewards whereas NPEC distance tends to saturate. For example, the NPEC distance from `Penalty` to other rewards lies in the very narrow $[0.98, 1.0]$ range, whereas EPIC uses the wider $[0.66, 1.0]$ range. See Figure B.1 for definitions of each reward. Distances are computed using tabular algorithms. We do not report confidence intervals since these algorithms are deterministic and exact up to floating point error.

Distance Metric	Wall-Clock Time	Environment Time Steps	Reward Queries	# of Seeds	95% CI Width Max	 Mean
EPIC Quick	17 s	8192	1.67×10^7	3	0.023 04	0.008 60
EPIC	6738 s	65 536	1.07×10^9	30	0.005 58	0.002 34
NPEC	29 769 s	7.50×10^7	7.50×10^7	3	0.315 91	0.066 20
ERC	1376 s	6.55×10^6	6.55×10^6	—	0.015 81	0.005 33
RL (PPO)	14 745 s	7.50×10^7	7.50×10^7	3	—	—

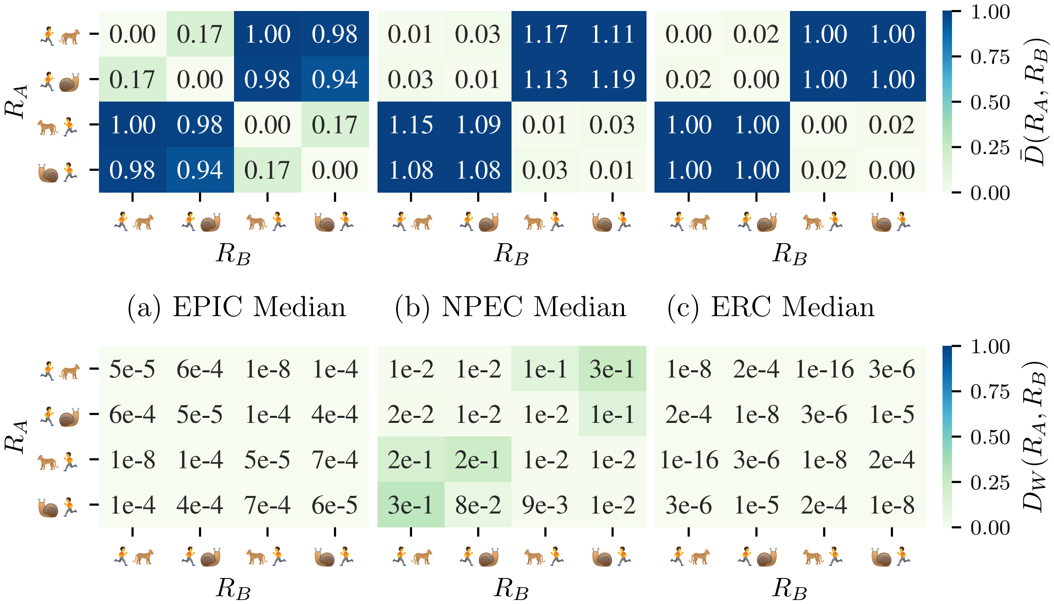
Table B.6: Time and resources taken by different metrics to perform 25 distance comparisons on PointMass, and the confidence interval widths obtained (smaller is better). Methods *EPIC*, *NPEC* and *ERC* correspond to Figures 4.2(a), (b) and (c) respectively. *EPIC Quick* is an abbreviated version with fewer samples. *RL (PPO)* is estimated from the time taken using PPO to train a single policy (16m:23s) until convergence (10^6 time steps). EPIC samples $N_M + N_V$ time steps from the environment and performs $N_M N_V$ reward queries. In *EPIC Quick*, $N_M = N_V = 4096$; in *EPIC*, $N_M = N_V = 302768$. Other methods query the reward once per environment time step.



(a) EPIC Median (b) NPEC Median (c) ERC Median

(d) EPIC CI Width (e) NPEC CI Width (f) ERC CI Width

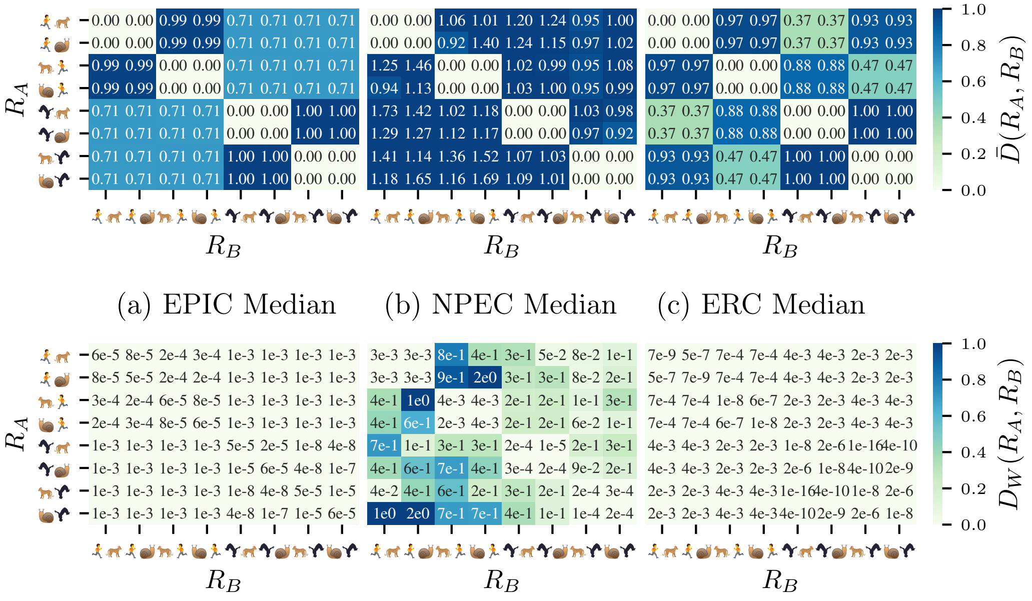
Figure B.3: Approximate distances between hand-designed reward functions in PointMass. The coverage distribution $\mathcal{D}$ is sampled from rollouts of a policy π_{uni} taking actions uniformly at random. **Key**: The agent has position $x \in \mathbb{R}$, velocity $\dot{x} \in \mathbb{R}$, and can accelerate $\ddot{x} \in \mathbb{R}$, producing future position $x' \in \mathbb{R}$. 🐌 quadratic penalty on control $\ddot{x}^2$, 🐆 no control penalty. S is Sparse$(x) = \mathbb{1}[|x| < 0.05]$, D is shaped Dense$(x, x') = $ Sparse$(x) + |x'| - |x|$, while M is Magnitude$(x) = -|x|$. **Confidence Interval (CI)**: 95% CI computed by bootstraping over 10 000 samples.



(a) EPIC Median (b) NPEC Median (c) ERC Median

(d) EPIC CI Width (e) NPEC CI Width (f) ERC CI Width

Figure B.4: Approximate distances between hand-designed reward functions in `HalfCheetah`. The coverage distribution $\mathcal{D}$ is sampled from rollouts of a policy π_{uni} taking actions uniformly at random. **Key**: 🏃 is a reward proportional to the change in center of mass and moving *forward* is rewarded when 🏃 to the right, and moving *backward* is rewarded when 🏃 to the left. 🐌 quadratic control penalty, 🐆 no control penalty. **Confidence Interval (CI)**: 95% CI computed by bootstraping over 10 000 samples.



(a) EPIC Median (b) NPEC Median (c) ERC Median

(d) EPIC CI Width (e) NPEC CI Width (f) ERC CI Width

Figure B.5: Approximate distances between hand-designed reward functions in `Hopper`. The coverage distribution $\mathcal{D}$ is sampled from rollouts of a policy π_{uni} taking actions uniformly at random. **Key**: 🏃 is a reward proportional to the change in center of mass and 🤸 is the backflip reward defined in Amodei, Christiano, and Ray [7, footnote]. Moving *forward* is rewarded when 🏃 or 🤸 is to the right, and moving *backward* is rewarded when 🏃 or 🤸 is to the left. 🐌 quadratic control penalty, 🐆 no control penalty. **Confidence Interval (CI)**: 95% CI computed by bootstraping over 10 000 samples.

Table B.7: Approximate distances of reward functions from the ground-truth (`GT`). We report the 95% bootstrapped lower and upper bounds, the mean, and a 95% bound on the relative error from the mean. Distances (1000× scale) use coverage distribution $\mathcal{D}$ from rollouts in the `PointMaze-Train` environment of: a uniform random policy $\boldsymbol{\pi_{\text{uni}}}$, an expert $\boldsymbol{\pi}^*$ and a **Mix**ture of these policies. $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{A}}$ are computed by marginalizing $\mathcal{D}$.

(a) 95% lower bound D^{LOW} of approximate distance.

Reward	**1000 ×** $\boldsymbol{D^{\text{LOW}}_{\text{EPIC}}}$			**1000 ×** $\boldsymbol{D^{\text{LOW}}_{\text{NPEC}}}$			**1000 ×** $\boldsymbol{D^{\text{LOW}}_{\text{ERC}}}$			**Episode Return**	
Function	π_{uni}	π^*	**Mix**	π_{uni}	π^*	**Mix**	π_{uni}	π^*	**Mix**	**Train**	**Test**
`GT`	0.03	0.02	<0.01	0.02	1.43	<0.01	0.00	0.00	0.00	−4.46	−5.82
`Regress`	35.5	33.6	26.0	1.22	38.8	0.33	9.94	90.2	2.42	−4.7	−5.63
`Pref`	68.3	100	56.6	7.02	1239	9.25	24.7	358	19.5	−5.26	−4.88
`AIRL SO`	570	519	402	734	1645	424	547	521	238	−27.3	−22.7
`AIRL SA`	774	930	894	956	723	952	802	720	963	−29.9	−28
`Mirage`	3.49	0.02	381	0.17	4.03	481	25.8	<0.01	162	−28.4	−26.2

(b) Mean approximate distance $\overline{D}$. Results are the same as Table 4.2.

Reward	**1000 ×** $\boldsymbol{\overline{D}_{\text{EPIC}}}$			**1000 ×** $\boldsymbol{\overline{D}_{\text{NPEC}}}$			**1000 ×** $\boldsymbol{\overline{D}_{\text{ERC}}}$			**Episode Return**	
Function	π_{uni}	π^*	**Mix**	π_{uni}	π^*	**Mix**	π_{uni}	π^*	**Mix**	**Train**	**Test**
`GT`	0.06	0.05	0.04	0.04	3.17	0.01	0.00	0.00	0.00	−5.19	−6.59
`Regress`	35.8	33.7	26.1	1.42	38.9	0.35	9.99	90.7	2.43	−5.47	−6.3
`Pref`	68.7	100	56.8	8.51	1333	9.74	24.9	360	19.6	−5.57	−5.04
`AIRL SO`	572	520	404	817	2706	488	549	523	240	−27.3	−22.7
`AIRL SA`	776	930	894	1067	2040	1039	803	722	964	−30.7	−29
`Mirage`	17.0	0.05	397	0.68	6.30	597	35.3	<0.01	166	−30.4	−29.1

(c) 95% upper bound D^{UP} of approximate distance.

Reward	**1000 × $\boldsymbol{D^{\mathrm{UP}}_{\mathrm{EPIC}}}$**			**1000 × $\boldsymbol{D^{\mathrm{UP}}_{\mathrm{NPEC}}}$**			**1000 × $\boldsymbol{D^{\mathrm{UP}}_{\mathrm{ERC}}}$**			**Episode Return**	
Function	π_{uni}	π^*	**Mix**	π_{uni}	π^*	**Mix**	π_{uni}	π^*	**Mix**	**Train**	**Test**
`GT`	0.09	0.07	0.07	0.06	6.14	0.01	<0.01	<0.01	<0.01	−6.04	−7.14
`Regress`	36.1	33.7	26.2	1.53	39.1	0.37	10.0	91.2	2.44	−6.26	−6.83
`Pref`	69.1	101	57.1	10.0	1432	10.1	25.2	361	19.7	−5.9	−5.22
`AIRL SO`	574	520	407	982	3984	532	551	526	242	−27.3	−22.8
`AIRL SA`	779	930	895	1241	4378	1124	805	724	964	−31.7	−29.8
`Mirage`	35.2	0.09	414	1.66	10.8	821	45.4	<0.01	171	−32	−31.4

(d) Relative 95% confidence interval $D^{\mathrm{REL}} = \left|\max\left(\frac{\text{Upper}}{\text{Mean}} - 1, 1 - \frac{\text{Lower}}{\text{Mean}}\right)\right|$ in percent. The population mean is contained within $\pm D^{\mathrm{REL}}\%$ of the sample mean in Table B.7b with 95% probability.

Reward	**$\boldsymbol{D^{\mathrm{REL}}_{\mathrm{EPIC}}}$%**			**$\boldsymbol{D^{\mathrm{REL}}_{\mathrm{NPEC}}}$%**			**$\boldsymbol{D^{\mathrm{REL}}_{\mathrm{ERC}}}$%**			**Episode Return**	
Function	π_{uni}	π^*	**Mix**	π_{uni}	π^*	**Mix**	π_{uni}	π^*	**Mix**	**Train**	**Test**
`GT`	50.0	62.5	80.0	61.8	94.0	29.7	inf	inf	inf	0.16	0.12
`Regress`	0.81	0.14	0.40	14.2	0.42	7.48	0.53	0.55	0.57	0.14	0.11
`Pref`	0.61	0.14	0.44	17.5	7.49	5.02	0.90	0.48	0.48	0.06	0.04
`AIRL SO`	0.38	0.08	0.67	20.2	47.2	13.2	0.34	0.40	0.69	<0.01	<0.01
`AIRL SA`	0.35	0.02	0.08	16.3	115	8.42	0.23	0.26	0.04	0.03	0.04
`Mirage`	108	65.5	4.17	142	70.9	37.5	28.5	0.55	2.66	0.07	0.10

Table B.8: Approximate distances of reward functions from the ground-truth (`GT`) under pathological coverage distributions. We report the 95% bootstrapped lower and upper bounds, the mean, and a 95% bound on the relative error from the mean. Distances (1000× scale) use four different coverage distributions $\mathcal{D}$. $\boldsymbol{\sigma}$ independently samples states, actions and next states from the marginal distributions of rollouts from the uniform random policy π_{uni} in the `PointMaze-Train` environment. **Ind** independently samples the components of states and next states from $\mathcal{N}(0, 1)$, and actions from $U[-1, 1]$. **Jail** consists of rollouts of π_{uni} restricted to a small 0.09 × 0.09 "jail" square that excludes the goal state 0.5 distance away. $\boldsymbol{\pi}_{\textbf{bad}}$ are rollouts in `PointMaze-Train` of a policy that goes to the corner opposite the goal state. $\boldsymbol{\sigma}$ and **Ind** are not supported by ERC since they do not produce complete episodes.

(a) 95% lower bound D^{LOW} of approximate distance.

Reward	$\mathbf{1000 \times \boldsymbol{D}^{\textbf{LOW}}_{\textbf{EPIC}}}$				$\mathbf{1000 \times \boldsymbol{D}^{\textbf{LOW}}_{\textbf{NPEC}}}$				$\mathbf{1000 \times \boldsymbol{D}^{\textbf{LOW}}_{\textbf{ERC}}}$	
Function	$\boldsymbol{\sigma}$	**Ind**	**Jail**	$\boldsymbol{\pi}_{\textbf{bad}}$	$\boldsymbol{\sigma}$	**Ind**	**Jail**	$\boldsymbol{\pi}_{\textbf{bad}}$	**Jail**	$\boldsymbol{\pi}_{\textbf{bad}}$
`Regress`	127	398	705	205	87.6	590	2433	898	809	456
`Pref`	146	433	462	349	97.4	632	661	221	372	332
`AIRL SO`	570	541	712	710	697	821	957	621	751	543
`AIRL SA`	768	628	558	669	720	960	940	2355	428	753
`Mirage`	9.22	0.03	<0.01	0.02	0.41	0.05	11.2	31.3	<0.01	0.02

(b) Mean approximate distance $\overline{D}$. Results are the same as Table 4.2.

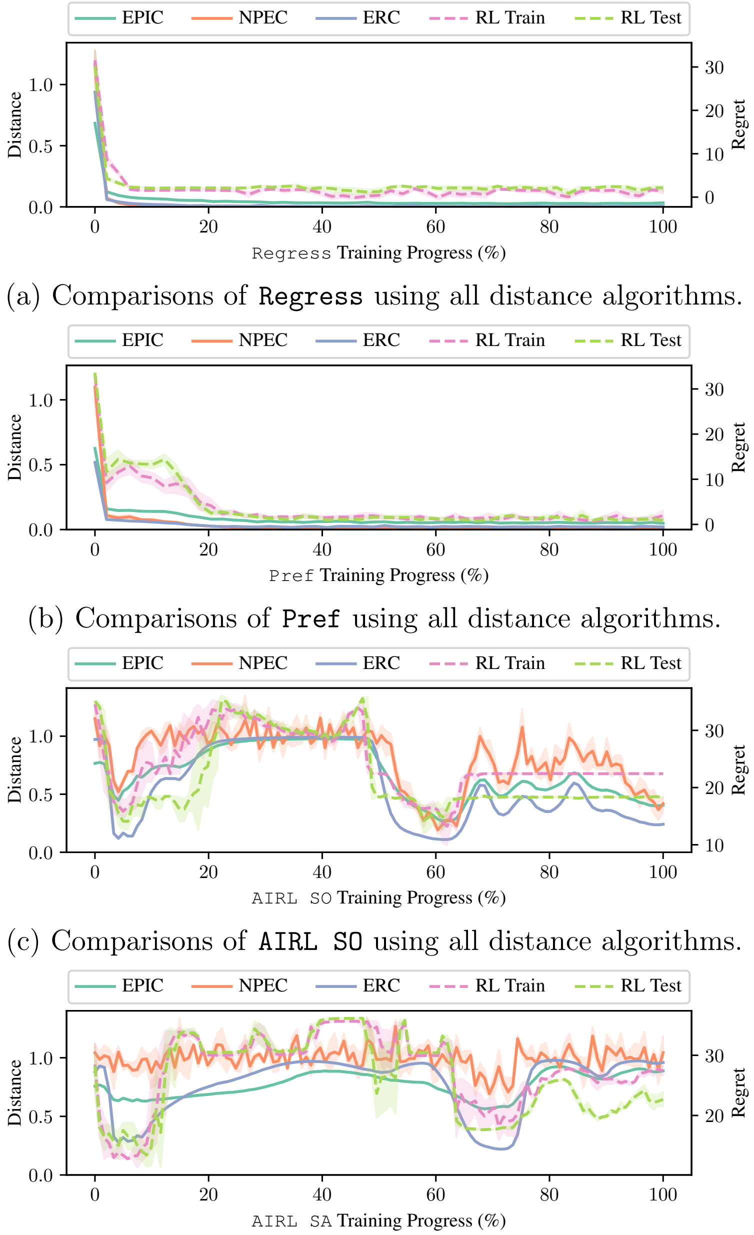
Reward	$\mathbf{1000 \times \overline{\boldsymbol{D}}_{\textbf{EPIC}}}$				$\mathbf{1000 \times \overline{\boldsymbol{D}}_{\textbf{NPEC}}}$				$\mathbf{1000 \times \overline{\boldsymbol{D}}_{\textbf{ERC}}}$	
Function	$\boldsymbol{\sigma}$	**Ind**	**Jail**	$\boldsymbol{\pi}_{\textbf{bad}}$	$\boldsymbol{\sigma}$	**Ind**	**Jail**	$\boldsymbol{\pi}_{\textbf{bad}}$	**Jail**	$\boldsymbol{\pi}_{\textbf{bad}}$
`Regress`	128	398	705	206	97.2	591	2549	921	810	458
`Pref`	147	433	463	349	117	633	683	237	374	333
`AIRL SO`	573	541	713	710	826	823	988	852	753	545
`AIRL SA`	771	628	558	669	859	962	964	2694	430	754
`Mirage`	42.4	0.06	0.03	0.05	1.41	0.25	18.3	39	<0.01	0.02

(c) 95% upper bound D^{UP} of approximate distance.

Reward	$\mathbf{1000 \times D^{UP}_{EPIC}}$				$\mathbf{1000 \times D^{UP}_{NPEC}}$				$\mathbf{1000 \times D^{UP}_{ERC}}$	
Function	$\boldsymbol{\sigma}$	**Ind**	**Jail**	$\boldsymbol{\pi_{\mathrm{bad}}}$	$\boldsymbol{\sigma}$	**Ind**	**Jail**	$\boldsymbol{\pi_{\mathrm{bad}}}$	**Jail**	$\boldsymbol{\pi_{\mathrm{bad}}}$
`Regress`	129	398	706	206	106	593	2654	948	812	460
`Pref`	148	433	464	349	132	635	705	265	376	335
`AIRL SO`	576	541	713	710	939	825	1047	1021	755	547
`AIRL SA`	774	628	559	669	1015	963	981	3012	432	756
`Mirage`	85.9	0.09	0.05	0.09	3.25	0.46	28.6	45.3	<0.01	0.02

(d) Relative 95% confidence interval $D^{\mathrm{REL}} = \left|\max\left(\frac{\mathrm{Upper}}{\mathrm{Mean}} - 1, 1 - \frac{\mathrm{Lower}}{\mathrm{Mean}}\right)\right|$ in percent. The population mean is contained within $\pm D^{\mathrm{REL}}\%$ of the sample mean in Table B.8b with 95% probability.

Reward	$\mathbf{D^{REL}_{EPIC}\%}$				$\mathbf{D^{REL}_{NPEC}\%}$				$\mathbf{D^{REL}_{ERC}\%}$	
Function	$\boldsymbol{\sigma}$	**Ind**	**Jail**	$\boldsymbol{\pi_{\mathrm{bad}}}$	$\boldsymbol{\sigma}$	**Ind**	**Jail**	$\boldsymbol{\pi_{\mathrm{bad}}}$	**Jail**	$\boldsymbol{\pi_{\mathrm{bad}}}$
`Regress`	0.79	0.01	0.08	0.06	9.81	0.24	4.54	2.89	0.18	0.42
`Pref`	0.76	<0.01	0.16	0.07	16.9	0.35	3.35	11.7	0.47	0.48
`AIRL SO`	0.48	<0.01	0.07	0.02	15.6	0.28	6.01	27.1	0.23	0.38
`AIRL SA`	0.38	<0.01	0.13	0.03	18.2	0.22	2.47	12.6	0.45	0.23
`Mirage`	103	50	80	83.3	131	85.4	56.4	19.7	0.54	0.55



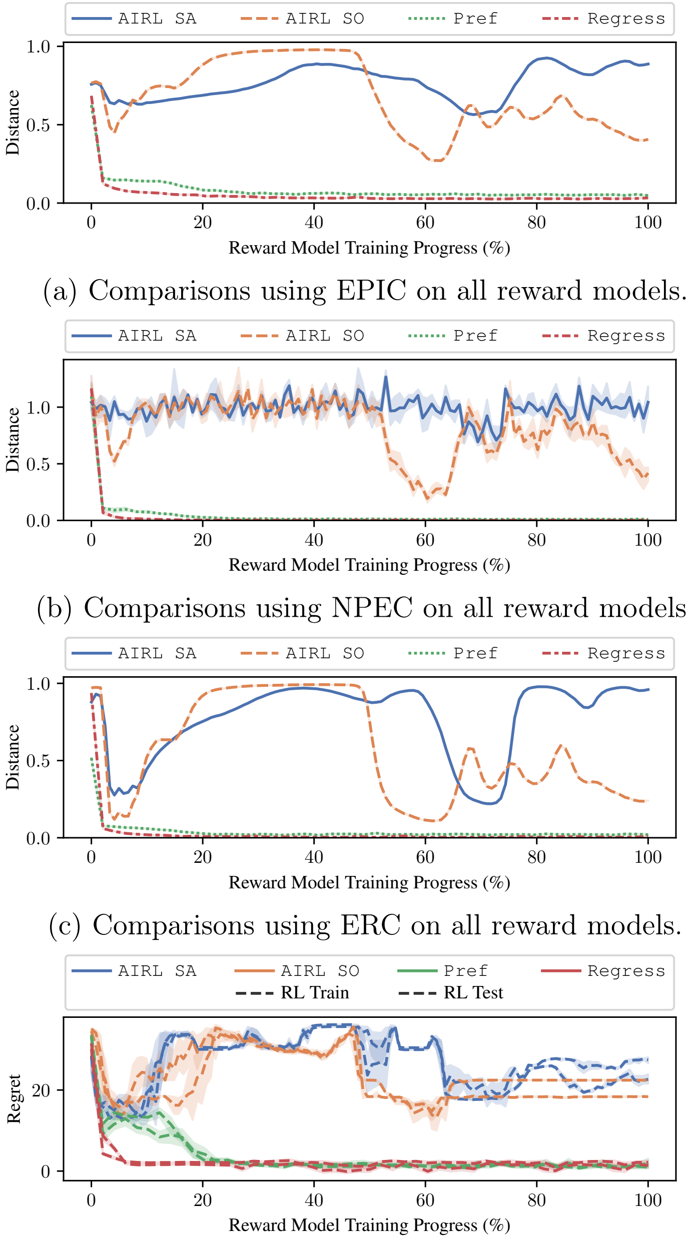
(a) Comparisons of `Regress` using all distance algorithms.

(b) Comparisons of `Pref` using all distance algorithms.

(c) Comparisons of `AIRL SO` using all distance algorithms.

(d) Comparisons of `AIRL SA` using all distance algorithms.

Figure B.6: Distance of reward checkpoints from the ground-truth in `PointMaze` and policy regret for reward checkpoints during reward model training. Each point evaluates a reward function checkpoint from a single seed. EPIC, NPEC and ERC distance use the `Mixture` distribution. Regret is computed by running RL on the checkpoint. The shaded region represents the bootstrapped 95% confidence interval for the distance or regret at that checkpoint, calculated following Section B.1.3.



(a) Comparisons using EPIC on all reward models.

(b) Comparisons using NPEC on all reward models.

(c) Comparisons using ERC on all reward models.

(d) Comparisons using Episode Return on all reward models.

Figure B.7: Distance of reward checkpoints from the ground-truth in `PointMaze` and policy regret for reward checkpoints during reward model training. Each point evaluates a reward function checkpoint from a single seed. EPIC, NPEC and ERC distance use the `Mixture` distribution. Regret is computed by running RL on the checkpoint. The shaded region represents the bootstrapped 95% confidence interval for the distance or regret at that checkpoint, calculated following Section B.1.3.

B.3 Proofs

B.3.1 Equivalent-Policy Invariant Comparison (EPIC) pseudometric

Proposition 4.2.3 (The Canonically Shaped Reward is Invariant to Shaping)**.** *Let $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be a reward function and $\Phi : \mathcal{S} \to \mathbb{R}$ a potential function. Let $\gamma \in [0, 1]$ be a discount rate, and $\mathcal{D}_{\mathcal{S}} \in \Delta(\mathcal{S})$ and $\mathcal{D}_{\mathcal{A}} \in \Delta(\mathcal{A})$ be distributions over states and actions. Let R' denote R shaped by Φ: $R'(s, a, s') = R(s, a, s') + \gamma\Phi(s') - \Phi(s)$. Then the canonically shaped R' and R are equal: $C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R') = C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)$.*

Proof. Let $s, a, s' \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$. Then by substituting in the definition of R' and using linearity of expectations:

$$\begin{aligned}
C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R')(s, a, s') &\triangleq R'(s, a, s') + \mathbb{E}\left[\gamma R'(s', A, S') - R'(s, A, S') - \gamma R'(S, A, S')\right] && \text{(B.8)} \\
&= (R(s, a, s') + \gamma\Phi(s') - \Phi(s)) && \text{(B.9)} \\
&\quad + \mathbb{E}\left[\gamma R(s', A, S') + \gamma^2\Phi(S') - \gamma\Phi(s')\right] \\
&\quad - \mathbb{E}\left[R(s, A, S') + \gamma\Phi(S') - \Phi(s)\right] \\
&\quad - \mathbb{E}\left[\gamma R(S, A, S') + \gamma^2\Phi(S') - \gamma\Phi(S)\right] \\
&= R(s, a, s') + \mathbb{E}\left[\gamma R(s', A, S') - R(s, A, S') - \gamma R(S, A, S')\right] && \text{(B.10)} \\
&\quad + (\gamma\Phi(s') - \Phi(s)) - \mathbb{E}\left[\gamma\Phi(s') - \Phi(s)\right] \\
&\quad + \mathbb{E}\left[\gamma^2\Phi(S') - \gamma\Phi(S')\right] - \mathbb{E}\left[\gamma^2\Phi(S') - \gamma\Phi(S)\right] \\
&= R(s, a, s') + \mathbb{E}\left[\gamma R(s', A, S') - R(s, A, S') - \gamma R(S, A, S')\right] && \text{(B.11)} \\
&\triangleq C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)(s, a, s'), && \text{(B.12)}
\end{aligned}$$

where the penultimate step uses $\mathbb{E}[\Phi(S')] = \mathbb{E}[\Phi(S)]$ since S and S' are identically distributed. □

Proposition 4.2.4. *Let $\mathcal{S}$ and $\mathcal{A}$ be finite, with $|\mathcal{S}| \geqslant 2$. Let $\mathcal{D}_{\mathcal{S}} \in \Delta(\mathcal{S})$ and $\mathcal{D}_{\mathcal{A}} \in \Delta(\mathcal{A})$. Let R, $\nu : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be reward functions, with $\nu(s, a, s') = \lambda\mathbb{I}[(s, a, s') = (x, u, x')]$, $\lambda \in \mathbb{R}$, $x, x' \in \mathcal{S}$, and $u \in \mathcal{A}$. Let $\Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)(s, a, s') = C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)(s, a, s') - R(s, a, s')$. Then,*

$$\left\|\Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R + \nu) - \Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)\right\|_\infty = \lambda\left(1 + \gamma\mathcal{D}_{\mathcal{S}}(x)\right)\mathcal{D}_{\mathcal{A}}(u)\mathcal{D}_{\mathcal{S}}(x'). \tag{4.2}$$

Proof. Observe that:

$$\Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)(s, a, s') = \mathbb{E}\left[\gamma R(s', A, S') - R(s, A, S') - \gamma R(S, A, S')\right], \tag{B.13}$$

where S and S' are random variables independently sampled from $\mathcal{D}_{\mathcal{S}}$, and A independently sampled from $\mathcal{D}_{\mathcal{A}}$.

Then:

$$\Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R + \nu) - \Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R) = \Phi_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(\nu). \tag{B.14}$$

Now:

$$
\begin{aligned}
\text{LHS} &\triangleq \left\|\Phi_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R+\nu) - \Phi_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R)\right\|_\infty && \text{(B.15)}\\
&= \max_{s,s'\in\mathcal{S}} \left|\mathbb{E}\left[\gamma\nu(s',A,S') - \nu(s,A,S') - \gamma\nu(S,A,S')\right]\right| && \text{(B.16)}\\
&= \max_{s,s'\in\mathcal{S}} |\lambda\left(\gamma\mathbb{I}[x=s']\mathcal{D}_\mathcal{A}(u)\mathcal{D}_\mathcal{S}(x')\right. && \text{(B.17)}\\
&- \left.\mathbb{I}[x=s]\mathcal{D}_\mathcal{A}(u)\mathcal{D}_\mathcal{S}(x') - \gamma\mathcal{D}_\mathcal{S}(x)\mathcal{D}_\mathcal{A}(u)\mathcal{D}_\mathcal{S}(x')\right)| && \text{(B.18)}\\
&= \max_{s,s'\in\mathcal{S}} \left|\lambda\mathcal{D}_\mathcal{A}(u)\mathcal{D}_\mathcal{S}(x')\left(\gamma\mathbb{I}[x=s'] - \mathbb{I}[x=s] - \gamma\mathcal{D}_\mathcal{S}(x)\right)\right| && \text{(B.19)}\\
&= \lambda\left(1+\gamma\mathcal{D}_\mathcal{S}(x)\right)\mathcal{D}_\mathcal{A}(u)\mathcal{D}_\mathcal{S}(x'), && \text{(B.20)}
\end{aligned}
$$

where the final step follows by substituting $s = x$ and $s' \neq x$ (using $|\mathcal{S}| \geqslant 2$). □

Lemma 4.2.6. *The Pearson distance D_ρ is a pseudometric. Moreover, let $a, b \in (0,\infty)$, $c, d \in \mathbb{R}$, and X, Y be random variables. Then it follows that $0 \leqslant D_\rho(aX + c, bY + d) = D_\rho(X,Y) \leqslant 1$.*

Proof. For a non-constant random variable V, define a standardized (zero mean and unit variance) version:

$$Z(V) = \frac{V - \mathbb{E}[V]}{\sqrt{\mathbb{E}\left[(V-\mathbb{E}[V])^2\right]}}. \tag{B.21}$$

The Pearson correlation coefficient on random variables A and B is equal to the expected product of these standardized random variables:

$$\rho(A,B) = \mathbb{E}\left[Z(A)Z(B)\right]. \tag{B.22}$$

Let W, X, Y be random variables.

Identity. Have $\rho(X,X) = 1$, so $D_\rho(X,X) = 0$.

Symmetry. Have $\rho(X,Y) = \rho(Y,X)$ by commutativity of multiplication, so $D_\rho(X,Y) = D_\rho(Y,X)$.

Triangle Inequality. For any random variables A, B:

$$
\begin{aligned}
\mathbb{E}\left[(Z(A)-Z(B))^2\right] &= \mathbb{E}\left[Z(A)^2 - 2Z(A)Z(B) + Z(B)^2\right] && \text{(B.23)}\\
&= \mathbb{E}\left[Z(A)^2\right] + \mathbb{E}\left[Z(B)^2\right] - 2\mathbb{E}\left[Z(A)Z(B)\right] && \text{(B.24)}\\
&= 2 - 2\mathbb{E}\left[Z(A)Z(B)\right] && \text{(B.25)}\\
&= 2\left(1-\rho(A,B)\right) && \text{(B.26)}\\
&= 4D_\rho(A,B)^2. && \text{(B.27)}
\end{aligned}
$$

So:

$$\begin{aligned}
4D_\rho(W,Y)^2 &= \mathbb{E}\left[(Z(W)-Z(Y))^2\right] && \text{(B.28)}\\
&= \mathbb{E}\left[(Z(W)-Z(X)+Z(X)-Z(Y))^2\right] && \text{(B.29)}\\
&= \mathbb{E}\left[(Z(W)-Z(X))^2\right]+\mathbb{E}\left[(Z(X)-Z(Y))^2\right] && \text{(B.30)}\\
&\quad + 2\mathbb{E}\left[(Z(W)-Z(X))\,(Z(X)-Z(Y))\right]\\
&= 4D_\rho(W,X)^2+4D_\rho(X,Y)^2+8\mathbb{E}\left[(Z(W)-Z(X))\,(Z(X)-Z(Y))\right]. && \text{(B.31)}
\end{aligned}$$

Since $\langle A,B\rangle = \mathbb{E}[AB]$ is an inner product over $\mathbb{R}$, it follows by the Cauchy-Schwarz inequality that $\mathbb{E}[AB] \leqslant \sqrt{\mathbb{E}[A^2]\mathbb{E}[B^2]}$. So:

$$\begin{aligned}
D_\rho(W,Y)^2 &\leqslant D_\rho(W,X)^2+D_\rho(X,Y)^2+2D_\rho(W,X)D_\rho(X,Y) && \text{(B.32)}\\
&= \left(D_\rho(W,X)+D_\rho(X,Y)\right)^2. && \text{(B.33)}
\end{aligned}$$

Taking the square root of both sides:

$$D_\rho(W,Y) \leqslant D_\rho(W,X)+D_\rho(X,Y), \tag{B.34}$$

as required.

Positive Affine Invariant and Bounded $D_\rho(aX+c,bY+d) = D_\rho(X,Y)$ is immediate from $\rho(X,Y)$ invariant to positive affine transformations. Have $-1 \leqslant \rho(X,Y) \leqslant 1$, so $0 \leqslant 1-\rho(X,Y) \leqslant 2$ thus $0 \leqslant D_\rho(X,Y) \leqslant 1$. □

Theorem 4.2.8. *The Equivalent-Policy Invariant Comparison distance is a pseudometric.*

Proof. The result follows from D_ρ being a pseudometric. Let R_A, R_B and R_C be reward functions mapping from transitions $\mathcal{S}\times\mathcal{A}\times\mathcal{S}$ to real numbers $\mathbb{R}$.

Identity. Have:

$$D_{\text{EPIC}}(R_A,R_A) = D_\rho\left(C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_A\right)(S,A,S'), C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_A\right)(S,A,S')\right) = 0, \tag{B.35}$$

since $D_\rho(X,X) = 0$.

Symmetry. Have:

$$\begin{aligned}
D_{\text{EPIC}}(R_A,R_B) &= D_\rho\left(C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_A\right)(S,A,S'), C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_B\right)(S,A,S')\right) && \text{(B.36)}\\
&= D_\rho\left(C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_B\right)(S,A,S'), C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_A\right)(S,A,S')\right) && \text{(B.37)}\\
&= D_{\text{EPIC}}(R_B,R_A), && \text{(B.38)}
\end{aligned}$$

since $D_\rho(X,Y) = D_\rho(Y,X)$.

Triangle Inequality. Have:

$$\begin{aligned}
D_{\text{EPIC}}(R_A,R_C) &= D_\rho\left(C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_A\right)(S,A,S'), C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_C\right)(S,A,S')\right) && \text{(B.39)}\\
&\leqslant D_\rho\left(C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_A\right)(S,A,S'), C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_B\right)(S,A,S')\right) && \text{(B.40)}\\
&+ D_\rho\left(C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_B\right)(S,A,S'), C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_C\right)(S,A,S')\right) && \text{(B.41)}\\
&= D_{\text{EPIC}}(R_A,R_B)+D_{\text{EPIC}}(R_B,R_C), && \text{(B.42)}
\end{aligned}$$

since $D_\rho(X,Z) \leqslant D_\rho(X,Y)+D_\rho(Y,Z)$. □

Theorem 4.2.9. *Let R_A, R'_A, $R_B, R'_B : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be reward functions such that $R'_A \equiv R_A$ and $R'_B \equiv R_B$. Then $0 \leqslant D_{\mathrm{EPIC}}(R'_A, R'_B) = D_{\mathrm{EPIC}}(R_A, R_B) \leqslant 1$.*

Proof. Since D_{EPIC} is defined in terms of D_ρ, the bounds $0 \leqslant D_{\mathrm{EPIC}}(R'_A, R'_B)$ and $D_{\mathrm{EPIC}}(R_A, R_B) \leqslant 1$ are immediate from the bounds in lemma 4.2.6.

Since $R'_A \equiv R_A$ and $R'_B \equiv R_B$, we can write for $X \in \{A, B\}$:

$$R^\lambda_X(s, a, s') = \lambda_X R_X(s, a, s'), \tag{B.43}$$

$$R'_X(s, a, s') = R^\lambda_X(s, a, s') + \gamma\Phi_X(s') - \Phi_X(s), \tag{B.44}$$

for some scaling factor $\lambda_X > 0$ and potential function $\Phi_X : \mathcal{S} \to \mathbb{R}$.

By proposition 4.2.3:

$$C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R'_X\right) = C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R^\lambda_X\right). \tag{B.45}$$

Moreover, since $C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R)$ is defined as an expectation over R and expectations are linear:

$$C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R^\lambda_X\right) = \lambda_X C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_X\right). \tag{B.46}$$

Unrolling the definition of D_{EPIC} and applying this result gives:

$$\begin{aligned} D_{\mathrm{EPIC}}(R'_A, R'_B) &= D_\rho\left(C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R'_A\right)(S, A, S'), C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R'_B\right)(S, A, S')\right) && \text{(B.47)}\\ &= D_\rho\left(\lambda_A C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_A\right)(S, A, S'), \lambda_B C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_B\right)(S, A, S')\right) && \text{eqs. B.45 and B.46}\\ &= D_\rho\left(C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_A\right)(S, A, S'), C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R_B\right)(S, A, S')\right) && \text{lemma 4.2.6}\\ &= D_{\mathrm{EPIC}}(R_A, R_B). \end{aligned}$$

□

B.3.2 Nearest Point in Equivalence Class (NPEC) premetric

Proposition B.3.1. *(1) $D_{L^p,\mathcal{D}}$ is a metric in L^p space, where functions f and g are identified if they agree almost everywhere on $\mathcal{D}$. (2) $D_{L^p,\mathcal{D}}$ is a pseudometric if functions are identified only if they agree at all points.*

Proof. (1) $D_{L^p,\mathcal{D}}$ is a metric in the L^p space since L^p is a norm in the L^p space, and $d(x, y) = \|x - y\|$ is always a metric. (2) As $f = g$ at all points implies $f = g$ almost everywhere, certainly $D_{L^p,\mathcal{D}}(R, R) = 0$. Symmetry and triangle inequality do not depend on identity so still hold. □

Proposition B.3.2 (Properties of D^U_{NPEC})**.** *Let $R_A, R_B, R_C : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be bounded reward functions, and $\lambda \geqslant 0$. Then D^U_{NPEC}:*

- ***Is invariant under $\equiv$ in source:***
  $D^U_{\mathrm{NPEC}}(R_A, R_C) = D^U_{\mathrm{NPEC}}(R_B, R_C)$ if $R_A \equiv R_B$.

- ***Invariant under scale-preserving $\equiv$ in target:***
  $D^U_{\mathrm{NPEC}}(R_A, R_B) = D^U_{\mathrm{NPEC}}(R_A, R_C)$ if $R_B - R_C \equiv \texttt{Zero}$.

- ***Scalable in target:*** $D^U_{\text{NPEC}}(R_A, \lambda R_B) = \lambda D^U_{\text{NPEC}}(R_A, R_B)$.
- ***Bounded:*** $D^U_{\text{NPEC}}(R_A, R_B) \leqslant D^U_{\text{NPEC}}(\texttt{Zero}, R_B)$.

Proof. We will show each case in turn.

Invariance under $\equiv$ in source

If $R_A \equiv R_B$, then:

$$\begin{aligned} D^U_{\text{NPEC}}(R_A, R_C) &\triangleq \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(R, R_C) && \text{(B.48)} \\ &= \inf_{R \equiv R_B} D_{L^p,\mathcal{D}}(R, R_C) && \text{(B.49)} \\ &\triangleq D^U_{\text{NPEC}}(R_B, R_C), && \text{(B.50)} \\ & && \text{(B.51)} \end{aligned}$$

since $R \equiv R_A$ if and only if $R \equiv R_B$ as $\equiv$ is an equivalence relation.

Invariance under scale-preserving $\equiv$ in target

If $R_B - R_C \equiv \texttt{Zero}$, then we can write $R_B(s,a,s') - R_C(s,a,s') = \gamma\Phi(s') - \Phi(s)$ for some potential function $\Phi : \mathcal{S} \to \mathbb{R}$. Define $f(R)(s,a,s') = R(s,a,s') - \gamma\Phi(s') + \Phi(s)$. Then for any reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$:

$$\begin{aligned} D_{L^p,\mathcal{D}}(R, R_B) &\triangleq \left(\mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}}\left[|R(s,a,s') - R_B(s,a,s')|^p\right]\right)^{1/p} \\ &= \left(\mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}}\left[|R(s,a,s') - (R_C(s,a,s') + \gamma\Phi(s') - \Phi(s))|^p\right]\right)^{1/p} \\ &= \left(\mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}}\left[|(R(s,a,s') - \gamma\Phi(s') + \Phi(s)) - R_C(s,a,s')|^p\right]\right)^{1/p} \\ &= \left(\mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}}\left[|f(R)(s,a,s') - R_C(s,a,s')|^p\right]\right)^{1/p} \\ &\triangleq D_{L^p,\mathcal{D}}(f(R), R_C), && \text{(B.52)} \end{aligned}$$

Crucially, note $f(R)$ is a bijection on the equivalence class $[R]$. Now, substituting this into the expression for the NPEC premetric:

$$\begin{aligned} D^U_{\text{NPEC}}(R_A, R_B) &\triangleq \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(R, R_B) \\ &= \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(f(R), R_C) && \text{eq. B.52} \\ &= \inf_{f(R) \equiv R_A} D_{L^p,\mathcal{D}}(f(R), R_C) && f \text{ bijection on } [R] \\ &= \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(R, R_C) && f \text{ bijection on } [R] \\ &\triangleq D^U_{\text{NPEC}}(R_A, R_C). && \text{(B.53)} \end{aligned}$$

Scalable in target

First, note that $D_{L^p,\mathcal{D}}$ is absolutely scalable in both arguments:

$$\begin{aligned}
D_{L^p,\mathcal{D}}(\lambda R_A, \lambda R_B) &\triangleq \left(\mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}} \left[\left| \lambda R_A(s,a,s') - \lambda R_B(s,a,s') \right|^p \right] \right)^{1/p} \\
&= \left(\mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}} \left[|\lambda|^p \left| R_A(s,a,s') - R_B(s,a,s') \right|^p \right] \right)^{1/p} && |\cdot| \text{ absolutely scalable} \\
&= \left(|\lambda|^p \mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}} \left[\left| R_A(s,a,s') - R_B(s,a,s') \right|^p \right] \right)^{1/p} && \mathbb{E} \text{ linear} \\
&= |\lambda| \left(\mathop{\mathbb{E}}_{s,a,s' \sim \mathcal{D}} \left[\left| R_A(s,a,s') - R_B(s,a,s') \right|^p \right] \right)^{1/p} \\
&\triangleq |\lambda| \, D_{L^p,\mathcal{D}}(R_A, R_B). && \text{(B.54)}
\end{aligned}$$

Now, for $\lambda > 0$, applying this to D^U_{NPEC}:

$$\begin{aligned}
D^U_{\text{NPEC}}(R_A, \lambda R_B) &\triangleq \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(R, \lambda R_B) && && \text{(B.55)} \\
&= \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(\lambda R, \lambda R_B) && R \equiv \lambda R && \text{(B.56)} \\
&= \inf_{R \equiv R_A} \lambda D_{L^p,\mathcal{D}}(R, R_B) && \text{eq. B.54} && \text{(B.57)} \\
&= \lambda \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(R, R_B) && && \text{(B.58)} \\
&\triangleq \lambda D^U_{\text{NPEC}}(R_A, R_B). && && \text{(B.59)}
\end{aligned}$$

In the case $\lambda = 0$, then:

$$\begin{aligned}
D^U_{\text{NPEC}}(R_A, \texttt{Zero}) &\triangleq \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(R, \texttt{Zero}) && && \text{(B.60)} \\
&= \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}\left(\frac{1}{2} R, \texttt{Zero} \right) && R \equiv \frac{1}{2} R && \text{(B.61)} \\
&= \inf_{R \equiv R_A} \frac{1}{2} D_{L^p,\mathcal{D}}(R, \texttt{Zero}) && && \text{(B.62)} \\
&= \frac{1}{2} \inf_{R \equiv R_A} D_{L^p,\mathcal{D}}(R, \texttt{Zero}) && && \text{(B.63)} \\
&= \frac{1}{2} D^U_{\text{NPEC}}(R_A, \texttt{Zero}). && && \text{(B.64)}
\end{aligned}$$

Rearranging, we have:

$$D^U_{\text{NPEC}}(R_A, \texttt{Zero}) = 0. \qquad \text{(B.65)}$$

Bounded

Let $d \triangleq D_{\text{NPEC}}(\texttt{Zero}, R_B)$. Then for any $\epsilon > 0$, there exists some potential function $\Phi : \mathcal{S} \to \mathbb{R}$ such that the L^p distance of the potential shaping $R(s, a, s') \triangleq \gamma\Phi(s) - \Phi(s)$ from R_B satisfies:

$$D_{L^p,\mathcal{D}}(R, R_B) \leqslant d + \epsilon. \tag{B.66}$$

Let $\lambda \in [0, 1]$. Define:

$$R'_\lambda(s, a, s') \triangleq \lambda R_A(s, a, s') + R(s, a, s'). \tag{B.67}$$

Now:

$$\begin{aligned}
D_{L^p,\mathcal{D}}(R'_\lambda, R) &\triangleq \left(\mathop{\mathbb{E}}_{s,a,s'\sim\mathcal{D}}\left[|R'_\lambda(s, a, s') - R(s, a, s')|^p\right]\right)^{1/p} && \text{(B.68)}\\
&= \left(\mathop{\mathbb{E}}_{s,a,s'\sim\mathcal{D}}\left[|\lambda R_A(s, a, s')|^p\right]\right)^{1/p} && \text{(B.69)}\\
&= \left(|\lambda|^p \mathop{\mathbb{E}}_{s,a,s'\sim\mathcal{D}}\left[|R_A(s, a, s')|^p\right]\right)^{1/p} && \text{(B.70)}\\
&= |\lambda| \left(\mathop{\mathbb{E}}_{s,a,s'\sim\mathcal{D}}\left[|R_A(s, a, s')|^p\right]\right)^{1/p} && \text{(B.71)}\\
&= |\lambda| D_{L^p,\mathcal{D}}(R_A, \texttt{Zero}). && \text{(B.72)}
\end{aligned}$$

Since R_A is bounded, $D_{L^p,\mathcal{D}}(R_A, \texttt{Zero})$ must be finite, so:

$$\lim_{\lambda\to 0^+} D_{L^p,\mathcal{D}}(R'_\lambda, R) = 0. \tag{B.73}$$

It follows that for any $\epsilon > 0$ there exists some $\lambda > 0$ such that:

$$D_{L^p,\mathcal{D}}(R'_\lambda, R) \leqslant \epsilon. \tag{B.74}$$

Note that $R_A \equiv R'_\lambda$ for all $\lambda > 0$. So:

$$\begin{aligned}
D_{\text{NPEC}}(R_A, R_B) &\leqslant D_{L^p,\mathcal{D}}(R'_\lambda, R_B) && && \text{(B.75)}\\
&\leqslant D_{L^p,\mathcal{D}}(R'_\lambda, R) + D_{L^p,\mathcal{D}}(R, R_B) && \text{prop. B.3.1} && \text{(B.76)}\\
&\leqslant \epsilon + (d + \epsilon) && \text{eq. B.66 and eq. B.74} && \text{(B.77)}\\
&= d + 2\epsilon. && && \text{(B.78)}
\end{aligned}$$

Since $\epsilon > 0$ can be made arbitrarily small, it follows:

$$D_{\text{NPEC}}(R_A, R_B) \leqslant d \triangleq D_{\text{NPEC}}(\texttt{Zero}, R_B), \tag{B.79}$$

completing the proof. □

Theorem 4.3.4. D_{NPEC} *is a premetric on the space of bounded reward functions. Moreover, let* $R_A, R_A', R_B, R_B' : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ *be bounded reward functions such that* $R_A \equiv R_A'$ *and* $R_B \equiv R_B'$*. Then* $0 \leqslant D_{\text{NPEC}}(R_A', R_B') = D_{\text{NPEC}}(R_A, R_B) \leqslant 1$.

Proof. We will first prove D_{NPEC} is a premetric, and then prove it is invariant and bounded.

Premetric

First, we will show that D_{NPEC} is a premetric.

Respects identity: $D_{\text{NPEC}}(R_A, R_A) = 0$

If $D^U_{\text{NPEC}}(\texttt{Zero}, R_A) = 0$ then $D_{\text{NPEC}}(R_A, R_A) = 0$ as required. Suppose from now on that $D^U_{\text{NPEC}}(R_A, R_A) \neq 0$. It follows from prop B.3.1 that $D_{L^p,\mathcal{D}}(R_A, R_A) = 0$. Since $X \equiv X$, 0 is an upper bound for $D^U_{\text{NPEC}}(R_A, R_A)$. By prop B.3.1 $D_{L^p,\mathcal{D}}$ is non-negative, so this is also a lower bound for $D^U_{\text{NPEC}}(R_A, R_A)$. So $D^U_{\text{NPEC}}(R_A, R_A) = 0$ and:

$$D_{\text{NPEC}}(R_A, R_A) = \frac{D^U_{\text{NPEC}}(R_A, R_A)}{D^U_{\text{NPEC}}(\texttt{Zero}, R_A)} = \frac{0}{D^U_{\text{NPEC}}(\texttt{Zero}, R_A)} = 0. \tag{B.80}$$

Well-defined: $D_{\text{NPEC}}(R_A, R_B) \geqslant 0$

By prop B.3.1, it follows that $D_{L^p,\mathcal{D}}(R, R_B) \geqslant 0$ for all reward functions $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$. Thus 0 is a lower bound for $\{D_{L^p,\mathcal{D}}(R, R_B) \mid R : \mathcal{S} \times \mathcal{A} \times \mathcal{S}\}$, and thus certainly a lower bound for $\{D_{L^p,\mathcal{D}}(R, Y) \mid R \equiv X\}$ for any reward function X. Since the infimum is the *greatest* lower bound, it follows that for any reward function X:

$$D^U_{\text{NPEC}}(X, R_B) \triangleq \inf_{R \equiv X} D_{L^p,\mathcal{D}}(R, R_B) \geqslant 0. \tag{B.81}$$

In the case that $D^U_{\text{NPEC}}(\texttt{Zero}, R_B) = 0$, then $D_{\text{NPEC}}(R_A, R_B) = 0$ which is non-negative. From now on, suppose that $D^U_{\text{NPEC}}(\texttt{Zero}, R_B) \neq 0$. The quotient of a non-negative value with a positive value is non-negative, so:

$$D_{\text{NPEC}}(R_A, R_B) = \frac{D^U_{\text{NPEC}}(R_A, R_B)}{D^U_{\text{NPEC}}(\texttt{Zero}, R_B)} \geqslant 0. \tag{B.82}$$

Invariant and Bounded

Since $R_B' \equiv R_B$, we have $R_B' - \lambda R_B \equiv \texttt{Zero}$ for some $\lambda > 0$. By proposition B.3.2, D^U_{NPEC} is invariant under scale-preserving $\equiv$ in target and scalable in target. That is, for any reward R:

$$D^U_{\text{NPEC}}(R, R_B') = D^U_{\text{NPEC}}(R, \lambda R_B) = \lambda D^U_{\text{NPEC}}(R, R_B). \tag{B.83}$$

In particular, $D^U_{\text{NPEC}}(\texttt{Zero}, R_B') = \lambda D^U_{\text{NPEC}}(\texttt{Zero}, R_B)$. As $\lambda > 0$, it follows that $D^U_{\text{NPEC}}(\texttt{Zero}, R_B') = 0 \iff D^U_{\text{NPEC}}(\texttt{Zero}, R_B) = 0$.

Suppose $D^U_{\text{NPEC}}(\texttt{Zero}, R_B) = 0$. Then $D_{\text{NPEC}}(R, R_B) = 0 = D_{\text{NPEC}}(R, R_B')$ for any reward R, so the result trivially holds. From now on, suppose $D^U_{\text{NPEC}}(\texttt{Zero}, R_B) \neq 0$.

By proposition B.3.2, D^U_{NPEC} is invariant to $\equiv$ in source. That is, $D^U_{\text{NPEC}}(R_A, R_B)$ $= D^U_{\text{NPEC}}(R_A', R_B)$, so:

$$D_{\text{NPEC}}(R_A', R_B) = \frac{D^U_{\text{NPEC}}(R_A', R_B)}{D^U_{\text{NPEC}}(\texttt{Zero}, R_B)} = \frac{D^U_{\text{NPEC}}(R_A, R_B)}{D^U_{\text{NPEC}}(\texttt{Zero}, R_B)} = D_{\text{NPEC}}(R_A, R_B). \tag{B.84}$$

By eq. (B.83):

$$D_{\text{NPEC}}(R_A, R'_B) = \frac{\lambda D^U_{\text{NPEC}}(R_A, R_B)}{\lambda D^U_{\text{NPEC}}(\texttt{Zero}, R_B)} = \frac{D^U_{\text{NPEC}}(R_A, R_B)}{D^U_{\text{NPEC}}(\texttt{Zero}, R_B)} = D_{\text{NPEC}}(R_A, R_B). \tag{B.85}$$

Since D_{NPEC} is a premetric it is non-negative. By the boundedness property of proposition B.3.2, $D^U_{\text{NPEC}}(R, R_B) \leqslant D^U_{\text{NPEC}}(\texttt{Zero}, R_B)$, so:

$$D_{\text{NPEC}}(R_A, R_B) = \frac{D^U_{\text{NPEC}}(R_A, R_B)}{D^U_{\text{NPEC}}(\texttt{Zero}, R_B)} \leqslant 1, \tag{B.86}$$

which completes the proof. □

Note when $D_{L^p,\mathcal{D}}$ is a metric, then $D_{\text{NPEC}}(X, Y) = 0$ if and only if $X = Y$.

Proposition B.3.3. *D_{NPEC} is* not *symmetric in the undiscounted case.*

Proof. We will provide a counterexample showing that D_{NPEC} is not symmetric.

Choose the state space $\mathcal{S}$ to be binary $\{0, 1\}$ and the actions $\mathcal{A}$ to be the singleton $\{0\}$. Choose the coverage distribution $\mathcal{D}$ to be uniform on $s \xrightarrow{0} s$ for $s \in \mathcal{S}$. Take $\gamma = 1$, i.e. undiscounted. Note that as the successor state is always the same as the start state, potential shaping has no effect on $D_{L^p,\mathcal{D}}$, so WLOG we will assume potential shaping is always zero.

Now, take $R_A(s) = 2s$ and $R_B(s) = 1$. Take $p = 1$ for the L^p distance. Observe that $D_{L^p,\mathcal{D}}(\texttt{Zero}, R_A) = \frac{1}{2}(|0| + |2|) = 1$ and $D_{L^p,\mathcal{D}}(\texttt{Zero}, R_B) = \frac{1}{2}(|1| + |1|) = 1$. Since potential shaping has no effect, $D^U_{\text{NPEC}}(\texttt{Zero}, R) = D_{L^p,\mathcal{D}}(\texttt{Zero}, R)$ and so $D_{\text{NPEC}}(\texttt{Zero}, R_A) = 1$ and $D_{\text{NPEC}}(\texttt{Zero}, R_B) = 1$.

Now:

$$D^U_{\text{NPEC}}(R_A, R_B) = \inf_{\lambda > 0} D_{L^p,\mathcal{D}}(\lambda R_A, R_B) \tag{B.87}$$

$$= \inf_{\lambda > 0} \frac{1}{2}(|1| + |2\lambda - 1|) \tag{B.88}$$

$$= \frac{1}{2}, \tag{B.89}$$

with the infimum attained at $\lambda = \frac{1}{2}$. But:

$$D^U_{\text{NPEC}}(R_B, R_A) = \inf_{\lambda > 0} D_{L^p,\mathcal{D}}(\lambda R_B, R_A) \tag{B.90}$$

$$= \inf_{\lambda > 0} \frac{1}{2} f(\lambda) \tag{B.91}$$

$$= \frac{1}{2} \inf_{\lambda > 0} f(\lambda), \tag{B.92}$$

where:

$$f(\lambda) = |\lambda| + |2 - \lambda|, \qquad \lambda > 0. \tag{B.93}$$

Note that:

$$f(\lambda) = \begin{cases} 2 & \lambda \in (0, 2], \\ 2\lambda - 2 & \lambda \in (2, \infty). \end{cases} \tag{B.94}$$

So $f(\lambda) \geqslant 2$ on all of its domain, thus:

$$D^U_{\text{NPEC}}(R_B, R_A) = 1. \tag{B.95}$$

Consequently:

$$D_{\text{NPEC}}(R_A, R_B) = \frac{1}{2} \neq 1 = D_{\text{NPEC}}(R_B, R_A), \tag{B.96}$$

so D_{NPEC} is not symmetric. □

B.3.3 Full Normalization Variant of EPIC

Previously, we used the Pearson distance D_ρ to compare the canonicalized rewards. Pearson distance is naturally invariant to scaling. An alternative is to explicitly normalize the canonicalized rewards, and then compare them using any metric over functions.

Definition B.3.4 (Normalized Reward)**.** Let $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be a bounded reward function. Let $\|\cdot\|$ be a norm on the vector space of reward functions over the real field. Then the normalized R is:

$$R^N(s, a, s') = \frac{R(s, a, s')}{\|R\|} \tag{B.97}$$

Note that $(\lambda R)^N = R^N$ for any $\lambda > 0$ as norms are absolutely homogeneous.

We say a reward is *standardized* if it has been canonicalized and then normalized.

Definition B.3.5 (Standardized Reward)**.** Let $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be a bounded reward function. Then the standardized R is:

$$R^S = \left(C_{\mathcal{D}_\mathcal{S}, \mathcal{D}_\mathcal{A}}(R)\right)^N. \tag{B.98}$$

Now, we can define a pseudometric based on the direct distance between the standardized rewards.

Definition B.3.6 (Direct Distance Standardized Reward)**.** Let $\mathcal{D}$ be some coverage distribution over transitions $s \xrightarrow{a} s'$. Let $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{A}$ be some distributions over states $\mathcal{S}$ and $\mathcal{A}$ respectively. Let S, A, S' be random variables jointly sampled from $\mathcal{D}$. The *Direct Distance Standardized Reward pseudometric* between two reward functions R_A and R_B is the L^p distance between their standardized versions over $\mathcal{D}$:

$$D_{\text{DDSR}}(R_A, R_B) = \frac{1}{2} D_{L^p, \mathcal{D}}\left(R^S_A(S, A, S'), R^S_B(S, A, S')\right), \tag{B.99}$$

where the normalization step, R^N, uses the L^p norm.

For brevity, we omit the proof that D_{DDSR} is a pseudometric, but this follows from $D_{L^p,\mathcal{D}}$ being a pseudometric in a similar fashion to theorem 4.2.8. Note it additionally is invariant to equivalence classes, similarly to EPIC.

Theorem B.3.7. *Let R_A, $R_A{}'$, R_B and $R_B{}'$ be reward functions mapping from transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ to real numbers $\mathbb{R}$ such that $R_A \equiv R_A{}'$ and $R_B \equiv R_B{}'$. Then:*

$$0 \leqslant D_{\mathrm{DDSR}}(R'_A, R'_B) = D_{\mathrm{DDSR}}(R_A, R_B) \leqslant 1. \tag{B.100}$$

Proof. The invariance under the equivalence class follows from R^S being invariant to potential shaping and scale in R. The non-negativity follows from $D_{L^p,\mathcal{D}}$ being a pseudometric. The upper bound follows from the rewards being normalized to norm 1 and the triangle inequality:

$$\begin{aligned} D_{\mathrm{DDSR}}(R_A, R_B) &= \frac{1}{2}\|R_A^S - R_B^S\| && \text{(B.101)} \\ &\leqslant \frac{1}{2}\left(\|R_A^S\| + \|R_B^S\|\right) && \text{(B.102)} \\ &= \frac{1}{2}(1+1) && \text{(B.103)} \\ &= 1. && \square \end{aligned}$$

Since both DDSR and EPIC are pseudometrics and invariant on equivalent rewards, it is interesting to consider the connection between them. In fact, under the L^2 norm, then DDSR recovers EPIC. First, we will show that canonical shaping centers the reward functions.

Lemma B.3.8 (The Canonically Shaped Reward is Mean Zero)**.** *Let R be a reward function mapping from transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ to real numbers $\mathbb{R}$. Then:*

$$\mathbb{E}\left[C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R\right)\left(S, A, S'\right)\right] = 0, \tag{B.104}$$

where S, S' are random variables sampled from $\mathcal{D}_\mathcal{S}$ and A is sampled from $\mathcal{D}_\mathcal{A}$.

Proof. Let X, U and X' be random variables that are independent of S, A and S' but identically distributed.

$$\begin{aligned} \text{LHS} &\triangleq \mathbb{E}\left[C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}\left(R\right)\left(S, A, S'\right)\right] && \text{(B.105)} \\ &= \mathbb{E}\left[R(S, A, S') + \gamma R(S', U, X') - R(S, U, X') - \gamma R(X, U, X')\right] && \text{(B.106)} \\ &= \mathbb{E}\left[R(S, A, S')\right] + \gamma\mathbb{E}\left[R(S', U, X')\right] - \mathbb{E}\left[R(S, U, X')\right] - \gamma\mathbb{E}\left[R(X, U, X')\right] && \text{(B.107)} \\ &= \mathbb{E}\left[R(S, U, X')\right] + \gamma\mathbb{E}\left[R(X, U, X')\right] - \mathbb{E}\left[R(S, U, X')\right] - \gamma\mathbb{E}\left[R(X, U, X')\right] && \text{(B.108)} \\ &= 0, && \text{(B.109)} \end{aligned}$$

where the penultimate step follows since A is identically distributed to U, and S' is identically distributed to X' and therefore to X. □

Theorem B.3.9. *D_{DDSR} with $p = 2$ is equivalent to D_{EPIC} when the canonicalization distribution $\mathcal{D}$ is equal to the coverage distribution $\mathcal{D}_{\mathcal{S}} \times \mathcal{D}_{\mathcal{A}} \times \mathcal{D}_{\mathcal{S}}$. Let R_A and R_B be reward functions mapping from transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ to real numbers $\mathbb{R}$. Then:*

$$D_{\text{DDSR}}(R_A, R_B) = D_{\text{EPIC}}(R_A, R_B). \tag{B.110}$$

Proof. Recall from the proof of lemma 4.2.6 that:

$$\begin{aligned} D_\rho(U, V) &= \frac{1}{2}\sqrt{\mathbb{E}\left[(Z(U) - Z(V))^2\right]} && \text{(B.111)} \\ &= \frac{1}{2}\left\|Z(U) - Z(V)\right\|_2, && \text{(B.112)} \end{aligned}$$

where $\|\cdot\|_2$ is the L^2 norm with respect to the coverage distribution $\mathcal{D}$ (treating the random variables as functions on a measure space), $\mathbb{E}[\cdot]$ is an expectation over $\mathcal{D}$, and $Z(U)$,$Z(V)$ are random variables centered (zero-mean) and rescaled (unit variance) with respect to $\mathcal{D}$. By lemma B.3.8, the canonically shaped reward functions are centered under the canonicalization distribution $\mathcal{D}_{\mathcal{S}} \times \mathcal{D}_{\mathcal{A}} \times \mathcal{D}_{\mathcal{S}}$. By our assumption that the canonicalization and coverage distributions are equal, canonically shaped reward functions are also centered on the coverage distribution. Normalization by the L^2 norm also ensures they have unit variance. Consequently:

$$\begin{aligned} D_{\text{EPIC}}(R_A, R_B) &= D_\rho\left(C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R_A\right)(S, A, S'), C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R_B\right)(S, A, S')\right) && \text{(B.113)} \\ &= \frac{1}{2}\left\|\left(C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R_A\right)(S, A, S')\right)^N - \left(C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}\left(R_B\right)(S, A, S')\right)^N\right\|_2 && \text{(B.114)} \\ &= \frac{1}{2}\left\|R_A^S(S, A, S') - R_B^S(S, A, S')\right\|_2 && \text{(B.115)} \\ &= \frac{1}{2}D_{L^p,\mathcal{D}}\left(R_A^S(S, A, S'), R_B^S(S, A, S')\right) && \text{(B.116)} \\ &= D_{\text{DDSR}}(R_A, R_B), && \text{(B.117)} \end{aligned}$$

completing the proof. □

B.4 Regret bound

In this section, we derive an upper bound on the regret in terms of the EPIC distance. Specifically, given two reward functions R_A and R_B with optimal policies π_A^* and π_B^*, we show that the regret (under reward R_A) of using policy π_B^* instead of a policy π_A^* is bounded by a function of $D_{\text{EPIC}}(R_A, R_B)$. First, in section B.4.1 we derive a bound for MDPs with finite state and action spaces. In section B.5 we then present an alternative bound for MDPs with arbitrary state and action spaces and Lipschitz reward functions. Finally, in section B.6 we show that in both cases the regret tends to 0 as $D_{\text{EPIC}}(R_A, R_B) \to 0$.

B.4.1 Discrete MDPs

We start in lemma B.4.1 by showing that L^2 distance upper bounds L^1 distance. Next, in lemma B.4.2 we show regret is bounded by the L^1 distance between reward functions using an argument similar to [154]. Then in lemma B.4.3 we relate regret bounds for standardized rewards R^S to the original reward R. Finally, in theorem 5.1.1 we use section B.3.3 to express D_{EPIC} in terms of the L^2 distance on standardized rewards, deriving a bound on regret in terms of the EPIC distance.

Lemma B.4.1. *Let $(\Omega, \mathcal{F}, P)$ be a probability space and $f : \Omega \to \mathbb{R}$ a measurable function whose absolute value raised to the n-th power for $n \in \{1, 2\}$ has a finite expectation. Then the L^1 norm of f is bounded above by the L^2 norm:*

$$\|f\|_1 \leqslant \|f\|_2. \tag{B.118}$$

Proof. Let X be a random variable sampled from P, and consider the variance of $f(X)$:

$$\begin{aligned}
\mathbb{E}\left[(|f(X)| - \mathbb{E}\left[|f(X)|\right])^2\right] &= \mathbb{E}\left[|f(X)|^2 - 2|f(X)|\mathbb{E}\left[|f(X)|\right] + \mathbb{E}\left[|f(X)|\right]^2\right] && \text{(B.119)}\\
&= \mathbb{E}\left[|f(X)|^2\right] - 2\mathbb{E}\left[|f(X)|\right]\mathbb{E}\left[|f(X)|\right] + \mathbb{E}\left[|f(X)|\right]^2 && \text{(B.120)}\\
&= \mathbb{E}\left[|f(X)|^2\right] - \mathbb{E}\left[|f(X)|\right]^2 && \text{(B.121)}\\
&\geqslant 0. && \text{(B.122)}
\end{aligned}$$

Rearranging terms, we have

$$\|f\|_2^2 = \mathbb{E}\left[|f(X)|^2\right] \geqslant \mathbb{E}\left[|f(X)|\right]^2 = \|f\|_1^2. \tag{B.123}$$

Taking the square root of both sides gives:

$$\|f\|_1 \leqslant \|f\|_2, \tag{B.124}$$

as required. □

Lemma B.4.2. *Let M be an MDP$\backslash R$ with finite state and action spaces $\mathcal{S}$ and $\mathcal{A}$. Let $R_A, R_B : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be rewards. Let π_A^* and π_B^* be policies optimal for rewards R_A and R_B in M. Let $\mathcal{D}_\pi(t, s_t, a_t, s_{t+1})$ denote the distribution over trajectories that policy π induces in M at time step t. Let $\mathcal{D}(s, a, s')$ be the (stationary) coverage distribution over transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ used to compute D_{EPIC}. Suppose that there exists some $K > 0$ such that $K\mathcal{D}(s_t, a_t, s'_{t+1}) \geqslant \mathcal{D}_\pi(t, s_t, a_t, s'_{t+1})$ for all time steps $t \in \mathbb{N}$, triples $s_t, a_t, s_{t+1} \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ and policies $\pi \in \{\pi_A^*, \pi_B^*\}$. Then the regret under R_A from executing π_B^* optimal for R_B instead of π_A^* is at most:*

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant \frac{2K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B). \tag{B.125}$$

Proof. Noting $G_{R_A}(\pi)$ is maximized when $\pi = \pi_A^*$, it is immediate that

$$\begin{aligned} G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) &= |G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*)| && \text{(B.126)} \\ &= |(G_{R_A}(\pi_A^*) - G_{R_B}(\pi_B^*)) + (G_{R_B}(\pi_B^*) - G_{R_A}(\pi_B^*))| && \text{(B.127)} \\ &\leqslant |G_{R_A}(\pi_A^*) - G_{R_B}(\pi_B^*)| + |G_{R_B}(\pi_B^*) - G_{R_A}(\pi_B^*)| \,. && \text{(B.128)} \end{aligned}$$

We will show that both these terms are bounded above by $\frac{K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B)$, from which the result follows.

First, we will show that for policy $\pi \in \{\pi_A^*, \pi_B^*\}$:

$$|G_{R_A}(\pi) - G_{R_B}(\pi)| \leqslant \frac{K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B). \tag{B.129}$$

Let T be the horizon of M. This may be infinite ($T = \infty$) when $\gamma < 1$; note since $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is bounded, so are R_A, R_B so the discounted infinite returns $G_{R_A}(\pi), G_{R_B}(\pi)$ converge (as do their differences). Writing $\tau = (s_0, a_0, s_1, a_1, \cdots)$, we have for any policy π:

$$\begin{aligned} \Delta &\triangleq |G_{R_A}(\pi) - G_{R_B}(\pi)| && \text{(B.130)} \\ &= \left| \mathbb{E}_{\tau\sim\mathcal{D}_\pi} \left[\sum_{t=0}^{T} \gamma^t \left(R_A(s_t, a_t, s_{t+1}) - R_B(s_t, a_t, s_{t+1}) \right) \right] \right| && \text{(B.131)} \\ &\leqslant \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}_\pi} \left[\sum_{t=0}^{T} \gamma^t \left| R_A(s_t, a_t, s_{t+1}) - R_B(s_t, a_t, s_{t+1}) \right| \right] && \text{(B.132)} \\ &= \sum_{t=0}^{T} \gamma^t \mathop{\mathbb{E}}_{s_t,a_t,s_{t+1}\sim\mathcal{D}_\pi} \left[\left| R_A(s_t, a_t, s_{t+1}) - R_B(s_t, a_t, s_{t+1}) \right| \right] && \text{(B.133)} \\ &= \sum_{t=0}^{T} \gamma^t \sum_{s_t,a_t,s_{t+1}\in\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mathcal{D}_\pi(t, s_t, a_t, s_{t+1}) \left| R_A(s_t, a_t, s_{t+1}) - R_B(s_t, a_t, s_{t+1}) \right| . && \text{(B.134)} \end{aligned}$$

Let $\pi \in \{\pi_A^*, \pi_B^*\}$. By assumption, $\mathcal{D}_\pi(t, s_t, a_t, s'_{t+1}) \leqslant K\mathcal{D}(s_t, a_t, s'_{t+1})$, so:

$$\begin{aligned} \Delta &\leqslant K \sum_{t=0}^{T} \gamma^t \sum_{s_t,a_t,s_{t+1}\in\mathcal{S}\times\mathcal{A}\times\mathcal{S}} \mathcal{D}(s_t, a_t, s_{t+1}) \left| R_A(s_t, a_t, s_{t+1}) - R_B(s_t, a_t, s_{t+1}) \right| && \text{(B.135)} \\ &= K \sum_{t=0}^{T} \gamma^t D_{L^1,\mathcal{D}}(R_A, R_B) && \text{(B.136)} \\ &= \frac{K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B), && \text{(B.137)} \end{aligned}$$

as required.

In particular, substituting $\pi = \pi_B^*$ gives:

$$|G_{R_B}(\pi_B^*) - G_{R_A}(\pi_B^*)| = |G_{R_A}(\pi_B^*) - G_{R_B}(\pi_B^*)| \leqslant \frac{K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B). \tag{B.138}$$

Rearranging gives:

$$G_{R_A}(\pi_B^*) \geqslant G_{R_B}(\pi_B^*) - \frac{K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B). \tag{B.139}$$

So certainly:

$$G_{R_A}(\pi_A^*) = \max_{\pi} G_{R_A}(\pi) \geqslant G_{R_B}(\pi_B^*) - \frac{K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B). \tag{B.140}$$

By a symmetric argument, substituting $\pi = \pi_A^*$ gives:

$$G_{R_B}(\pi_B^*) = \max_{\pi} G_{R_B}(\pi) \geqslant G_{R_A}(\pi_A^*) - \frac{K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B). \tag{B.141}$$

Eqs. B.140 and B.141 respectively give $G_{R_B}(\pi_B^*) - G_{R_A}(\pi_A^*) \leqslant \frac{K}{1-\gamma}$ and $G_{R_A}(\pi_A^*) - G_{R_B}(\pi_B^*) \leqslant \frac{K}{1-\gamma}$. Combining these gives:

$$|G_{R_A}(\pi_A^*) - G_{R_B}(\pi_B^*)| \leqslant \frac{K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B). \tag{B.142}$$

Substituting inequalities B.138 and B.142 into eq. B.128 yields the required result. □

Note that if $\mathcal{D} = \mathcal{D}_{\text{unif}}$, uniform over $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$, then $K \leqslant |\mathcal{S}|^2 |\mathcal{A}|$.

Lemma B.4.3. *Let M be an MDP$\backslash R$ with state and action spaces $\mathcal{S}$ and $\mathcal{A}$. Let $R_A, R_B : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be bounded rewards. Let π_A^* and π_B^* be policies optimal for rewards R_A and R_B in M. Suppose the regret under the standardized reward R_A^S from executing π_B^* instead of π_A^* is upper bounded by some $U \in \mathbb{R}$:*

$$G_{R_A^S}(\pi_A^*) - G_{R_A^S}(\pi_B^*) \leqslant U. \tag{B.143}$$

Additionally assume that the L^2 norm $||\cdot||_2$ in the standardization of R_A^S is taken with respect to the canonicalization distribution $\mathcal{D}_{\mathcal{S}} \times \mathcal{D}_{\mathcal{A}} \times \mathcal{D}_{\mathcal{S}}$. Then the regret under the original reward R_A is bounded by:

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant 4U \|R_A\|_2. \tag{B.144}$$

Proof. Recall that

$$R^S = \frac{C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)}{\|C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)\|_2}, \tag{B.145}$$

where $C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)$ is simply R shaped with some (bounded) potential Φ. It follows that:

$$\begin{aligned} G_{R^S}(\pi) &= \frac{1}{\|C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)\|_2} G_{C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)}(\pi) & \text{(B.146)} \\ &= \frac{1}{\|C_{\mathcal{D}_{\mathcal{S}},\mathcal{D}_{\mathcal{A}}}(R)\|_2} \left(G_R(\pi) - \mathbb{E}_{s_0 \sim \mu_0}\left[\Phi(s_0)\right] \right), & \text{(B.147)} \end{aligned}$$

where s_0 depends only on the initial state distribution μ_0. [*] Since s_0 does not depend on π, the terms cancel when taking the difference in returns:

$$G_{R_A^S}(\pi_A^*) - G_{R_A^S}(\pi_B^*) = \frac{1}{\|C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R_A)\|_2}\left(G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*)\right). \tag{B.148}$$

Combining this with eq B.143 gives

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant U\|C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R_A)\|_2. \tag{B.149}$$

Finally, we will bound $\|C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R_A)\|_2$ in terms of $\|R_A\|_2$, completing the proof. Recall:

$$C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R)(s,a,s') = R(s,a,s') + \mathbb{E}\left[\gamma R(s',A,S') - R(s,A,S') - \gamma R(S,A,S')\right], \tag{B.150}$$

where S and S' are random variables independently sampled from $\mathcal{D}_\mathcal{S}$ and A sampled from $\mathcal{D}_\mathcal{A}$. By the triangle inequality on the L^2 norm and linearity of expectations, we have:

$$\|C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R)\|_2 \leqslant \|R\|_2 + \gamma\|f\|_2 + \|g\|_2 + \gamma|c|, \tag{B.151}$$

where $f(s,a,s') = \mathbb{E}\left[R(s',A,S')\right]$, $g(s,a,s') = \mathbb{E}\left[R(s,A,S')\right]$ and $c = \mathbb{E}\left[R(S,A,S')\right]$. Letting X' be a random variable sampled from $\mathcal{D}_\mathcal{S}$ independently from S and S', have

$$\begin{aligned}
\|f\|_2^2 &= \mathbb{E}_{X'}\left[\mathbb{E}\left[R(X',A,S')\right]^2\right] && \text{(B.152)}\\
&\leqslant \mathbb{E}_{X'}\left[\mathbb{E}\left[R(X',A,S')^2\right]\right] && \text{(B.153)}\\
&= \mathbb{E}\left[R(X',A,S')^2\right] && \text{(B.154)}\\
&= \|R\|_2^2. && \text{(B.155)}
\end{aligned}$$

So $\|f\|_2 \leqslant \|R\|_2$ and, by an analogous argument, $\|g\|_2 \leqslant \|R\|_2$. Similarly

$$\begin{aligned}
|c| &= \left|\mathbb{E}\left[R(S,A,S')\right]\right| && \text{(B.156)}\\
&\leqslant \mathbb{E}\left[|R(S,A,S')|\right] && \text{(B.157)}\\
&= \|R\|_1 && \text{(B.158)}\\
&\leqslant \|R\|_2 \qquad \text{lemma B.4.1.} && \text{(B.159)}
\end{aligned}$$

Combining these results, we have

$$\|C_{\mathcal{D}_\mathcal{S},\mathcal{D}_\mathcal{A}}(R)\|_2 \leqslant 4\|R\|_2. \tag{B.160}$$

Substituting eq. B.160 into eq. B.149 yields:

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant 4U\|R_A\|_2, \tag{B.161}$$

as required. □

[*] In the finite-horizon case, there is also a term $\gamma^T\Phi(s_T)$, where s_T is the fixed terminal state. Since s_T is fixed, it also cancels in eq. B.148. This term can be neglected in the discounted infinite-horizon case as $\gamma^T\Phi(s_T) \to 0$ as $T \to \infty$ for any bounded Φ.

Theorem 5.1.1. *Let M be a γ-discounted MDP$\backslash R$ with finite state and action spaces $\mathcal{S}$ and $\mathcal{A}$. Let $R_A, R_B : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be rewards, and π_A^*, π_B^* be respective optimal policies. Let $\mathcal{D}_\pi(t, s_t, a_t, s_{t+1})$ denote the distribution over transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ induced by policy π at time t, $\mathcal{D}(s, a, s')$ be the coverage distribution used to compute D_{EPIC}, and $\mathcal{D}_{\mathcal{S}}(s), \mathcal{D}_{\mathcal{A}}(a)$ be the distributions defining the canonicalization in D_{EPIC}. Assume the coverage distribution is set equal to the canonicalization distribution: $\mathcal{D}(s, a, s') = \mathcal{D}_{\mathcal{S}}(s)\mathcal{D}_{\mathcal{A}}(a)\mathcal{D}_{\mathcal{S}}(s') \,\forall\, s, s' \in \mathcal{S}, a \in \mathcal{A}$. Suppose there exists $K > 0$ such that $K\mathcal{D}(s_t, a_t, s_{t+1}) \geqslant \mathcal{D}_\pi(t, s_t, a_t, s_{t+1})$ for all times $t \in \mathbb{N}$, triples $(s_t, a_t, s_{t+1}) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, and policies $\pi \in \{\pi_A^*, \pi_B^*\}$. Then, the regret under R_A from executing π_B^* instead of π_A^* is at most*

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant 16K\|R_A\|_2 \,(1-\gamma)^{-1}\, D_{\text{EPIC}}(R_A, R_B),$$

where $G_R(\pi)$ is the return of policy π under reward R, and the L^2 norm $||R_A||_2$ is taken with respect to the coverage distribution $\mathcal{D}$.

Proof. Recall from section B.3.3 that:

$$D_{\text{EPIC}}(R_A, R_B) = \frac{1}{2}\left\|R_A^S(S, A, S') - R_B^S(S, A, S')\right\|_2. \tag{B.162}$$

Applying lemma B.4.1 we obtain:

$$D_{L^1,\mathcal{D}}(R_A^S, R_B^S) = \left\|R_A^S(S, A, S') - R_B^S(S, A, S')\right\|_1 \leqslant 2D_{\text{EPIC}}(R_A, R_B). \tag{B.163}$$

Note that π_A^* is optimal for R_A^S and π_B^* is optimal for R_B^S since the set of optimal policies for R^S is the same as for R. Applying lemma B.4.2 and eq. B.163 gives

$$G_{R_A^S}(\pi_A^*) - G_{R_A^S}(\pi_B^*) \leqslant \frac{2K}{1-\gamma} D_{L^1,\mathcal{D}}(R_A^S, R_B^S) \leqslant \frac{4K}{1-\gamma} D_{\text{EPIC}}(R_A, R_B). \tag{B.164}$$

Since $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is bounded, R_A and R_B must be bounded, so we can apply lemma B.4.3, giving:

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant \frac{16K\|R_A\|_2}{1-\gamma} D_{\text{EPIC}}(R_A, R_B), \tag{B.165}$$

where the L^2 norm is taken with respect to the coverage distribution (since we assumed the coverage and canonicalization distributions to be equal). □

B.5 Lipschitz reward functions

In this section, we generalize the previous results to MDPs with continuous state and action spaces. The challenge is that even though the spaces may be continuous, the distribution $\mathcal{D}_{\pi_\star}$ induced by an optimal policy $\pi_\star$ may only have support on some measure zero set of transitions B. However, the expectation over a continuous distribution $\mathcal{D}$ is unaffected by the reward at any measure zero subset of points. Accordingly, the reward can be varied *arbitrarily* on transitions B – causing arbitrarily small or large regret – while leaving the EPIC distance fixed. To rule out this pathological case, we assume the rewards are Lipschitz smooth. This guarantees that if the expected difference between rewards is small on a given region, then all points in this region have bounded reward difference.

We start by defining a relaxation of the Wasserstein distance W_α in definition B.5.1. In lemma B.5.2 we then bound the expected value under distribution μ in terms of the expected value under alternative distribution ν plus $W_\alpha(\mu, \nu)$. Next, in lemma B.5.3 we bound the regret in terms of the L^1 distance between the rewards plus W_α; this is analogous to lemma B.4.2 in the finite case. Finally, in theorem B.5.4 we use the previous results to bound the regret in terms of the EPIC distance plus W_α.

Definition B.5.1. Let S be some set and let μ, ν be probability measures on S with finite first moment. We define the *relaxed Wasserstein distance* between μ and ν by:

$$W_\alpha(\mu, \nu) \triangleq \inf_{p \in \Gamma_\alpha(\mu,\nu)} \int \|x - y\| \, dp(x, y), \tag{B.166}$$

where $\Gamma_\alpha(\mu, \nu)$ is the set of probability measures on $S \times S$ satisfying for all $x, y \in S$:

$$\int_S p(x, y) dy = \mu(x), \tag{B.167}$$

$$\int_S p(x, y) dx \leqslant \alpha\nu(y). \tag{B.168}$$

Note that W_1 is equal to the (unrelaxed) Wasserstein distance (in the ℓ_1 norm).

Lemma B.5.2. *Let S be some set and let μ, ν be probability measures on S. Let $f : S \to \mathbb{R}$ be an L-Lipschitz function on the ℓ_1 norm $\|\cdot\|_1$. Then, for any $\alpha \geqslant 1$:*

$$\mathbb{E}_{X\sim\mu}\left[|f(X)|\right] \leqslant \alpha\mathbb{E}_{Y\sim\nu}\left[|f(Y)|\right] + LW_\alpha(\mu, \nu). \tag{B.169}$$

Proof. Let $p \in \Gamma_\alpha(\mu, \nu)$. Then:

$$\begin{aligned}
\mathbb{E}_{X\sim\mu}\left[|f(X)|\right] &\triangleq \int |f(x)| d\mu(x) && \text{definition of } \mathbb{E} \quad \text{(B.170)}\\
&= \int |f(x)| dp(x,y) && \mu \text{ is a marginal of } p \quad \text{(B.171)}\\
&\leqslant \int |f(y)| + L\left\|x-y\right\| dp(x,y) && f\ L\text{-Lipschitz} \quad \text{(B.172)}\\
&= \int |f(y)| dp(x,y) + L \int \left\|x-y\right\| dp(x,y) && \text{(B.173)}\\
&= \int |f(y)| \int p(x,y) dx dy + L \int \left\|x-y\right\| dp(x,y) && \text{(B.174)}\\
&\leqslant \int |f(y)| \alpha\nu(y) dy + L \int \left\|x-y\right\| dp(x,y) && \text{eq. B.168} \quad \text{(B.175)}\\
&= \alpha\mathbb{E}_{Y\sim\nu}\left[|f(Y)|\right] + L \int \left\|x-y\right\| dp(x,y) && \text{definition of } \mathbb{E}. \quad \text{(B.176)}
\end{aligned}$$

Since this holds for all choices of p, we can take the infimum of both sides, giving:

$$\begin{aligned}
\mathbb{E}_{X\sim\mu}\left[|f(X)|\right] &\leqslant \alpha\mathbb{E}_{Y\sim\nu}\left[|f(Y)|\right] + L \inf_{p\in\Gamma_\alpha(\mu,\nu)} \int \left\|x-y\right\| dp(x,y) && \text{(B.177)}\\
&= \alpha\mathbb{E}_{Y\sim\nu}\left[|f(Y)|\right] + L W_\alpha(\mu,\nu). && \square
\end{aligned}$$

Lemma B.5.3. *Let M be an MDP\R with state and action spaces $\mathcal{S}$ and $\mathcal{A}$. Let $R_A, R_B : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ be L-Lipschitz, bounded rewards on the ℓ_1 norm $\left\|\cdot\right\|_1$. Let π_A^* and π_B^* be policies optimal for rewards R_A and R_B in M. Let $\mathcal{D}_{\pi,t}(s_t, a_t, s_{t+1})$ denote the distribution over trajectories that policy π induces in M at time step t. Let $\mathcal{D}(s, a, s')$ be the (stationary) coverage distribution over transitions $\mathcal{S}\times\mathcal{A}\times\mathcal{S}$ used to compute D_{EPIC}. Let $\alpha \geqslant 1$, and let $B_\alpha(t) = \max_{\pi\in\{\pi_A^*,\pi_B^*\}} W_\alpha\left(\mathcal{D}_{\pi,t}, \mathcal{D}\right)$. Then the regret under R_A from executing π_B^* optimal for R_B instead of π_A^* is at most:*

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant \frac{2\alpha}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B) + 4L\sum_{t=0}^{\infty}\gamma^t B_\alpha(t). \tag{B.178}$$

Proof. By the same argument as lemma B.4.2 up to eq. B.134, we have for any policy π:

$$\left|G_{R_A}(\pi) - G_{R_B}(\pi)\right| \leqslant \sum_{t=0}^{\infty}\gamma^t D_{L^1,\mathcal{D}_{\pi,t}}(R_A, R_B). \tag{B.179}$$

Let $f(s, a, s') = R_A(s, a, s') - R_B(s, a, s')$, and note f is $2L$-Lipschitz and bounded since R_A and R_B are both L-Lipschitz and bounded. Now, by lemma B.5.2, letting $\mu = \mathcal{D}_{\pi,t}$ and $\nu = \mathcal{D}$, we have:

$$D_{L^1,\mathcal{D}_{\pi,t}}(R_A, R_B) \leqslant \alpha D_{L^1,\mathcal{D}}(R_A, R_B) + 2LW_\alpha(\mathcal{D}_{\pi,t}, \mathcal{D}). \tag{B.180}$$

So, for $\pi \in \{\pi_A^*, \pi_B^*\}$, it follows that

$$|G_{R_A}(\pi) - G_{R_B}(\pi)| \leqslant \frac{\alpha}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B) + 2L \sum_{t=0}^{\infty} \gamma^t B_\alpha(t). \tag{B.181}$$

By the same argument as for eq. B.138 to B.142 in lemma B.4.2, it follows that

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant \frac{2\alpha}{1-\gamma} D_{L^1,\mathcal{D}}(R_A, R_B) + 4L \sum_{t=0}^{\infty} \gamma^t B_\alpha(t), \tag{B.182}$$

completing the proof. □

Theorem B.5.4. *Let M be an MDP$\backslash$R with state and action spaces $\mathcal{S}$ and $\mathcal{A}$. Let $R_A, R_B : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be bounded, L-Lipschitz rewards on the ℓ_1 norm $\|\cdot\|_1$. Let π_A^* and π_B^* be policies optimal for rewards R_A and R_B in M. Let $\mathcal{D}_\pi(t, s_t, a_t, s_{t+1})$ denote the distribution over trajectories that policy π induces in M at time step t.*

Let $\mathcal{D}(s, a, s')$ be the (stationary) coverage distribution over transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ used to compute D_{EPIC}. Let $\mathcal{D}_\mathcal{S}(s)$ and $\mathcal{D}_\mathcal{A}(a)$ be the distributions over $\mathcal{S}$ and $\mathcal{A}$ used to canonicalize reward functions in the computation of D_{EPIC}. Assume the coverage and canonicalization distributions to be equal: $\mathcal{D}(s, a, s') = \mathcal{D}_\mathcal{S}(s)\mathcal{D}_\mathcal{A}(a)\mathcal{D}_\mathcal{S}(s') \, \forall \, s, s' \in \mathcal{S}, a \in \mathcal{A}$.

Let $\alpha \geqslant 1$, and let $B_\alpha(t) = \max_{\pi \in \pi_A^*, \pi_B^*} W_\alpha(\mathcal{D}_{\pi,t}, \mathcal{D})$. Then the regret under R_A from executing π_B^* optimal for R_B instead of π_A^* is at most:

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant 16 \|R_A\|_2 \left(\frac{\alpha}{1-\gamma} D_{\text{EPIC}}(R_A, R_B) + L \sum_{t=0}^{\infty} \gamma^t B_\alpha(t) \right). \tag{B.183}$$

Proof. The proof for theorem 5.1.1 holds in the general setting up to eq. B.163. Applying lemma B.5.3 to eq. B.163 gives

$$\begin{aligned} G_{R_A^S}(\pi_A^*) - G_{R_A^S}(\pi_B^*) &\leqslant \frac{2\alpha}{1-\gamma} D_{L^1,\mathcal{D}}(R_A^S, R_B^S) + 4L \sum_{t=0}^{\infty} \gamma^t B_\alpha(t) && \text{(B.184)} \\ &\leqslant \frac{4\alpha}{1-\gamma} D_{\text{EPIC}}(R_A, R_B) + 4L \sum_{t=0}^{\infty} \gamma^t B_\alpha(t). && \text{(B.185)} \end{aligned}$$

Applying lemma B.4.3 yields

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant 16 \|R_A\|_2 \left(\frac{\alpha}{1-\gamma} D_{\text{EPIC}}(R_A, R_B) + L \sum_{t=0}^{\infty} \gamma^t B_\alpha(t) \right), \tag{B.186}$$

as required. □

B.6 Limiting behavior of regret

Throughout this section we assume the coverage distribution to be set equal to the canonicalization distribution.

The regret bound for finite MDPs, Theorem 5.1.1, directly implies that, as EPIC distance tends to 0, the regret also tends to 0. By contrast, our regret bound in theorem B.5.4 for (possibly continuous) MDPs with Lipschitz reward functions includes the relaxed Wasserstein distance W_α as an additive term. At first glance, it might therefore appear possible for the regret to be positive even with a zero EPIC distance. However, in this section we will show that in fact the regret tends to 0 as $D_{\text{EPIC}}(R_A, R_B) \to 0$ in the Lipschitz case as well as the finite case.

We show in lemma B.6.1 that if the expectation of a non-negative function over a totally bounded measurable metric space M tends to zero under one distribution with adequate support, then it also tends to zero under all other distributions. For example, taking M to be a hypercube in Euclidean space with the Lebesque measure satisfies these assumptions. We conclude in theorem B.6.2 by showing the regret tends to 0 as the EPIC distance tends to 0.

Lemma B.6.1. *Let $M = (S, d)$ be a totally bounded metric space, where $d(x, y) = \|x - y\|$. Let (S, A, μ) be a measure space on S with the Borel σ-algebra A and measure μ. Let $p, q \in \Delta(S)$ be probability density functions on S. Let $\delta > 0$ such that $p(s) \geqslant \delta$ for all $s \in S$. Let $f_n : S \to \mathbb{R}$ be a sequence of L-Lipschitz functions on norm $\|\cdot\|$. Suppose $\lim_{n\to\infty} \mathbb{E}_{X\sim p}\left[|f_n(X)|\right] = 0$. Then $\lim_{n\to\infty} \mathbb{E}_{Y\sim q}\left[|f_n(Y)|\right] = 0$.*

Proof. Since M is totally bounded, for each $r > 0$ there exists a finite collection of open balls in S of radius r whose union contains M. Let $B_r(c) = \{s \in S \mid \|s - c\| < r\}$, the open ball of radius r centered at c. Let $C(r)$ denote some finite collection of $Q(r)$ open balls:

$$C(r) = \{B_r(c_{r,n}) \mid n \in \{1, \cdots, Q(r)\}\}, \tag{B.187}$$

such that $\bigcup_{B\in C(r)} B = S$.

It is possible for some balls $B_r(c_n)$ to have measure zero, $\mu(B_r(c_n)) = 0$, such as if S contains an isolated point c_n. Define $P(r)$ to be the subset of $C(r)$ with positive measure:

$$P(r) = \{B \in C(r) \mid \mu(B) > 0\}, \tag{B.188}$$

and let $p_{r,1}, \cdots, p_{r,Q'(r)}$ denote the centers of the balls in P. Since $P(r)$ is a finite collection, it must have a minimum measure:

$$\alpha(r) = \min_{B\in P} \mu(B). \tag{B.189}$$

Moreover, by construction of P, $\alpha(r) > 0$.

Let $S'(r)$ be the union only over balls of positive measure:

$$S'(r) = \bigcup_{B\in P(r)} B. \tag{B.190}$$

Now, let $D(r) = S \backslash S'(r)$, comprising the (finite number of) measure zero balls in $C(r)$. Since measures are countably additive, it follows that $D(r)$ is itself measure zero: $\mu\left(D(r)\right) = 0$. Consequently:

$$\int_S g(s) d\mu = \int_{S'(r)} g(s) d\mu, \tag{B.191}$$

for any measurable function $g : S \to \mathbb{R}$.

Since $\lim_{n\to\infty} \mathbb{E}_{X\sim p}\left[|f_n(X)|\right] = 0$, for all $r > 0$ there exists some $N_r \in \mathbb{N}$ such that for all $n \geqslant N_r$:

$$\mathbb{E}_{X\sim p}\left[|f_n(X)|\right] < \delta L r \alpha(r). \tag{B.192}$$

By Lipschitz continuity, for any $s, s' \in S$:

$$|f_n(s')| \geqslant |f_n(s)| - L\left\|s' - s\right\|. \tag{B.193}$$

In particular, since any point $s \in S'(r)$ is at most r distance from some ball center $p_{r,i}$, then $|f_n(p_{r,i})| \geqslant |f_n(s)| - Lr$. So if there exists $s \in S'(r)$ such that $|f_n(s)| \geqslant 3Lr$, then there must exist a ball center $p_{r,i}$ with $|f_n(p_{r,i})| \geqslant 2Lr$. Then for any point $s' \in B_r(p_{r,i})$:

$$|f_n(s')| \geqslant |f_n(p_{r,i})| - Lr \geqslant Lr. \tag{B.194}$$

Now, we have:

$$\begin{aligned}
\mathbb{E}_{X\sim p}\left[|f_n(X)|\right] &\triangleq \int_{s\in S} |f_n(s)| p(s) d\mu(s) && && \text{(B.195)}\\
&= \int_{s\in S'(r)} |f_n(s)| p(s) d\mu(s) && \text{eq. B.191} && \text{(B.196)}\\
&\geqslant \int_{s\in B_r(p_{r,i})} |f_n(s)| p(s) d\mu(s) && \text{non-negativity of } |f_n(s)| && \text{(B.197)}\\
&\geqslant \delta \int_{s\in B_r(p_{r,i})} |f_n(s)| d\mu(s) && p(s) \geqslant \delta && \text{(B.198)}\\
&\geqslant \delta \cdot Lr \int_{s\in B_r(p_{r,i})} 1 d\mu(s) && \text{eq. B.194} && \text{(B.199)}\\
&= \delta L r \mu(B_r(p_{r,i})) && \text{integrating w.r.t. } \mu && \text{(B.200)}\\
&\geqslant \delta L r \alpha(r) && \alpha(r) \text{ minimum of } \mu(B_r(p_{r,i})). && \text{(B.201)}
\end{aligned}$$

But this contradicts eq. B.192, and so can only hold if $n < N_r$. It follows that for all $n \geqslant N_r$ and $s \in S'(r)$, we have $|f_n(s)| < 3Lr$, and so in particular:

$$\mathbb{E}_{Y\sim q}\left[|f_n(Y)|\right] < 3Lr. \tag{B.202}$$

Let $\epsilon > 0$. Choose $r = \frac{\epsilon}{3L}$. Then for all $n \geqslant N_r$, $\mathbb{E}_{Y\sim q}\left[|f_n(Y)|\right] < \epsilon$. It follows that:

$$\lim_{n\to\infty} \mathbb{E}_{Y\sim q}\left[|f_n(Y)|\right] = 0, \tag{B.203}$$

completing the proof. □

Theorem B.6.2. *Let M be an MDP$\backslash R$ with state and action spaces $\mathcal{S}$ and $\mathcal{A}$. Let $R_A, R_B : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ be bounded rewards on some norm $\|\cdot\|$ on $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$. Let π_A^* and π_B^* be policies optimal for rewards R_A and R_B in M. Let $\mathcal{D}_\pi(t, s_t, a_t, s_{t+1})$ denote the distribution over trajectories that policy π induces in M at time step t.*

Let $\mathcal{D}(s, a, s')$ be the (stationary) coverage distribution over transitions $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ used to compute D_{EPIC}. Let $\mathcal{D}_{\mathcal{S}}(s)$ and $\mathcal{D}_{\mathcal{A}}(a)$ be the distributions over $\mathcal{S}$ and $\mathcal{A}$ used to canonicalize reward functions in the computation of D_{EPIC}. Assume the coverage and canonicalization distributions to be equal: $\mathcal{D}(s, a, s') = \mathcal{D}_{\mathcal{S}}(s)\mathcal{D}_{\mathcal{A}}(a)\mathcal{D}_{\mathcal{S}}(s') \,\forall\, s, s' \in \mathcal{S}, a \in \mathcal{A}$.

Suppose that either:

1. Discrete*: $\mathcal{S}$ and $\mathcal{A}$ are* discrete*. Moreover, suppose that there exists some $K > 0$ such that $K\mathcal{D}(s_t, a_t, s'_{t+1}) \geqslant \mathcal{D}_\pi(t, s_t, a_t, s'_{t+1})$ for all time steps $t \in \mathbb{N}$, triples $s_t, a_t, s_{t+1} \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ and policies $\pi \in \{\pi_A^*, \pi_B^*\}$.*

2. Lipschitz*: $(\mathcal{S} \times \mathcal{A} \times \mathcal{S}, d)$ is a totally bounded measurable metric space where $d(x, y) = \|x - y\|$. Moreover, R_A and R_B are L-*Lipschitz *on $\|\cdot\|$. Furthermore, suppose there exists some $\delta > 0$ such that $\mathcal{D}(s, a, s') \geqslant \delta$ for all $s, a, s' \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, and that $\mathcal{D}_\pi(t, s_t, a_t, s_{t+1})$ is a non-degenerate probability density function (i.e. no single point has positive measure).*

Then as $D_{\text{EPIC}}(R_A, R_B) \to 0$, $G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \to 0$.

Proof. In case *(1) Discrete*, by theorem 5.1.1:

$$G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \leqslant \frac{16K\|R_A\|_2}{1-\gamma} D_{\text{EPIC}}(R_A, R_B). \tag{B.204}$$

Moreover, by optimality of π_A^* we have $0 \leqslant G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*)$. So by the squeeze theorem, as $D_{\text{EPIC}}(R_A, R_B) \to 0$, $G_{R_A}(\pi_A^*) - G_{R_A}(\pi_B^*) \to 0$.

From now on, suppose we are in case *(2) Lipschitz.* By the same argument as lemma B.4.2 up to eq. B.134, we have for any policy π:

$$\left|G_{R_A^S}(\pi) - G_{R_B^S}(\pi)\right| \leqslant \sum_{t=0}^{\infty} \gamma^t D_{L^1, \mathcal{D}_{\pi,t}}(R_A^S, R_B^S). \tag{B.205}$$

Applying lemma B.4.3 we have:

$$|G_{R_A}(\pi) - G_{R_B}(\pi)| \leqslant 4 \|R_A\|_2 \sum_{t=0}^{\infty} \gamma^t D_{L^1, \mathcal{D}_{\pi,t}}(R_A^S, R_B^S). \tag{B.206}$$

By equation B.163, we know that $D_{L^1,\mathcal{D}}(R_A^S, R_B^S) \to 0$ as $D_{\text{EPIC}}(R_A, R_B) \to 0$. By lemma B.6.1, we know that $D_{L^1,\mathcal{D}_{\pi,t}}(R_A^S, R_B^S) \to 0$ as $D_{L^1,\mathcal{D}}(R_A^S, R_B^S) \to 0$. So we can conclude that as $D_{\text{EPIC}}(R_A, R_B) \to 0$:

$$|G_{R_A}(\pi) - G_{R_B}(\pi)| \to 0, \tag{B.207}$$

as required. □

Appendix C

Deferred content from Chapter 6

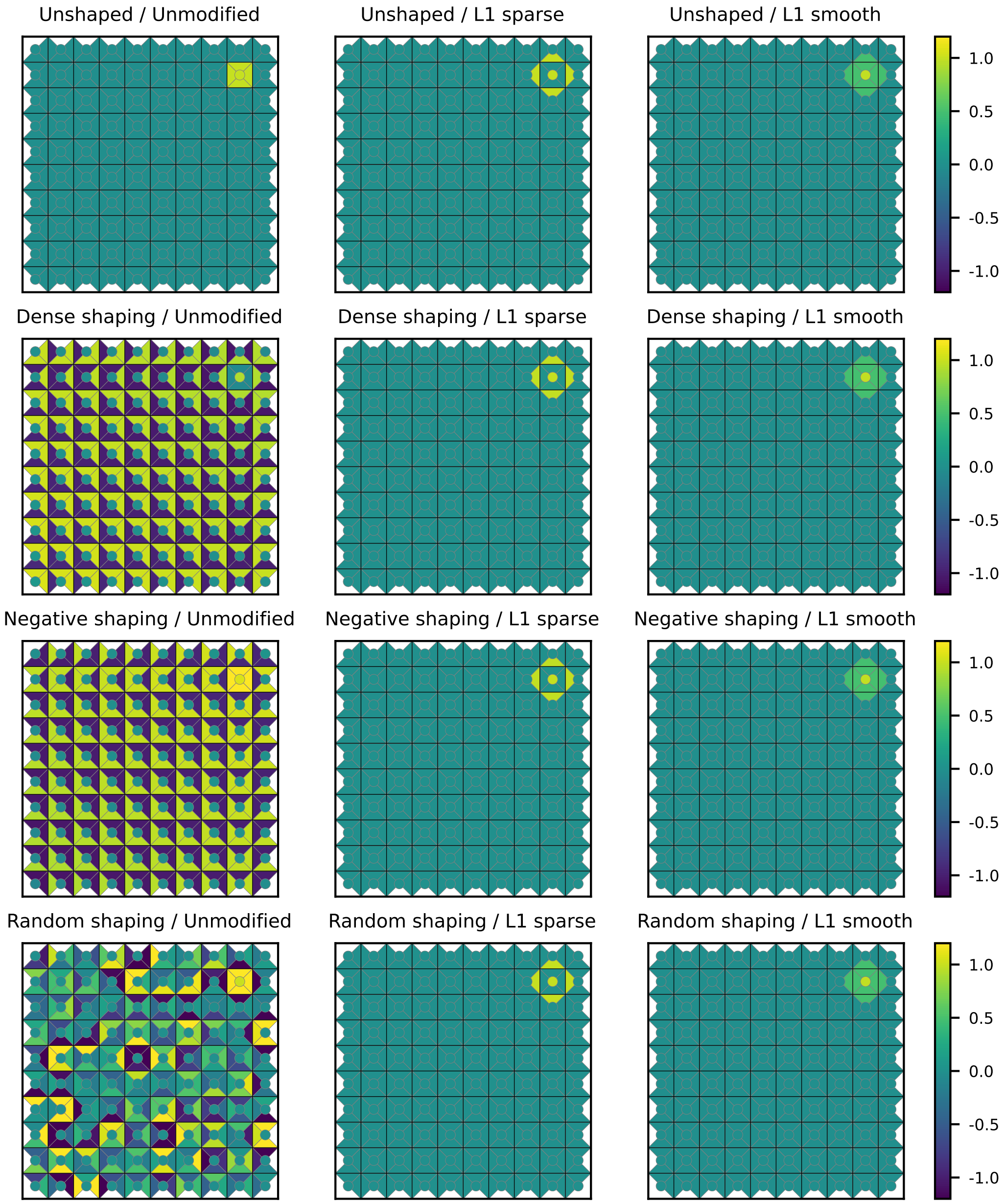


Figure C.1: `Goal` reward preprocessed with L^1 versions of the sparsity and smoothness cost functions. The results are very similar to those in Figure 6.1. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

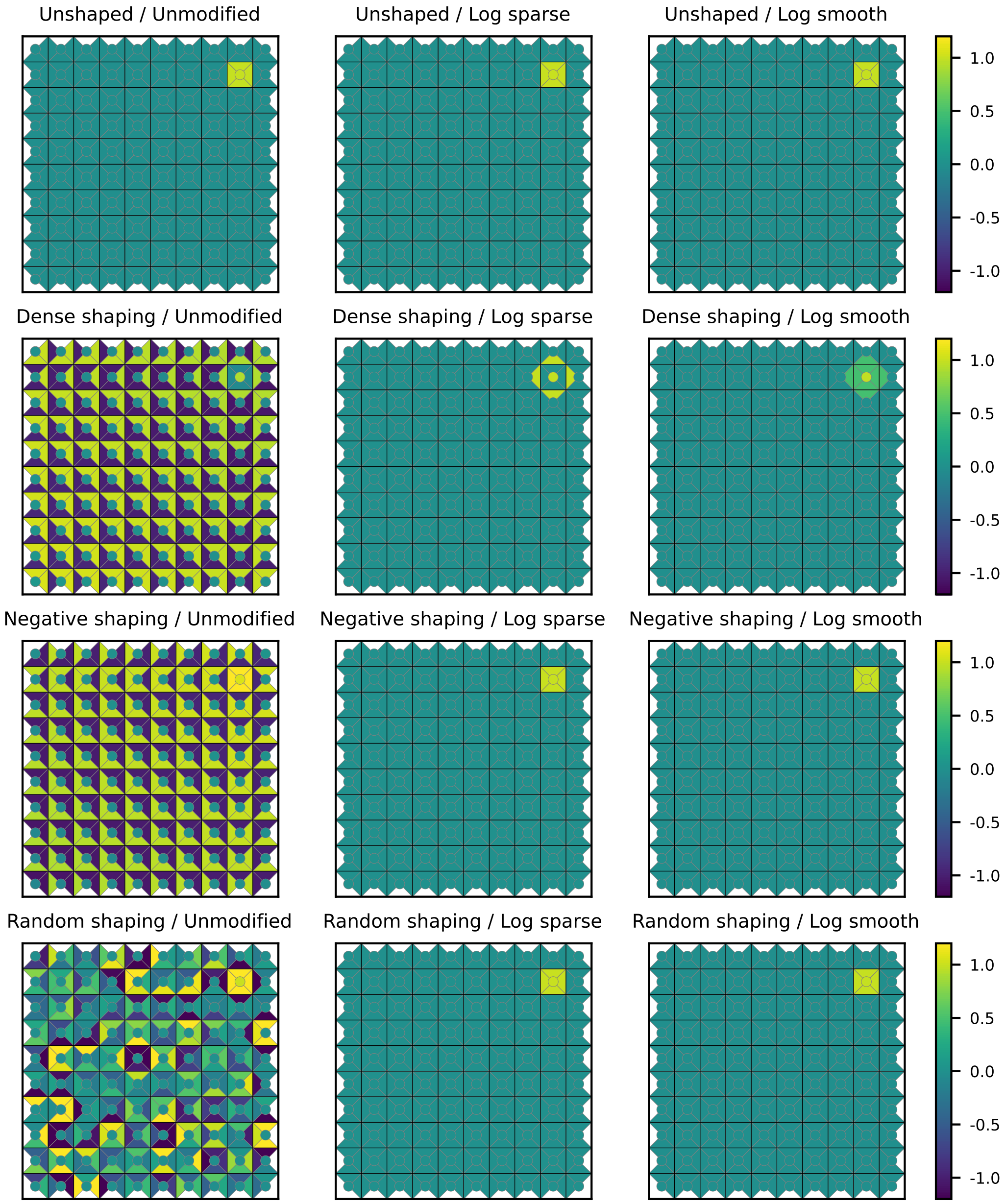


Figure C.2: `Goal` reward preprocessed with logarithmic versions of the sparsity and smoothness cost functions. Again, the results are qualitatively similar to those in Figure 6.1. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

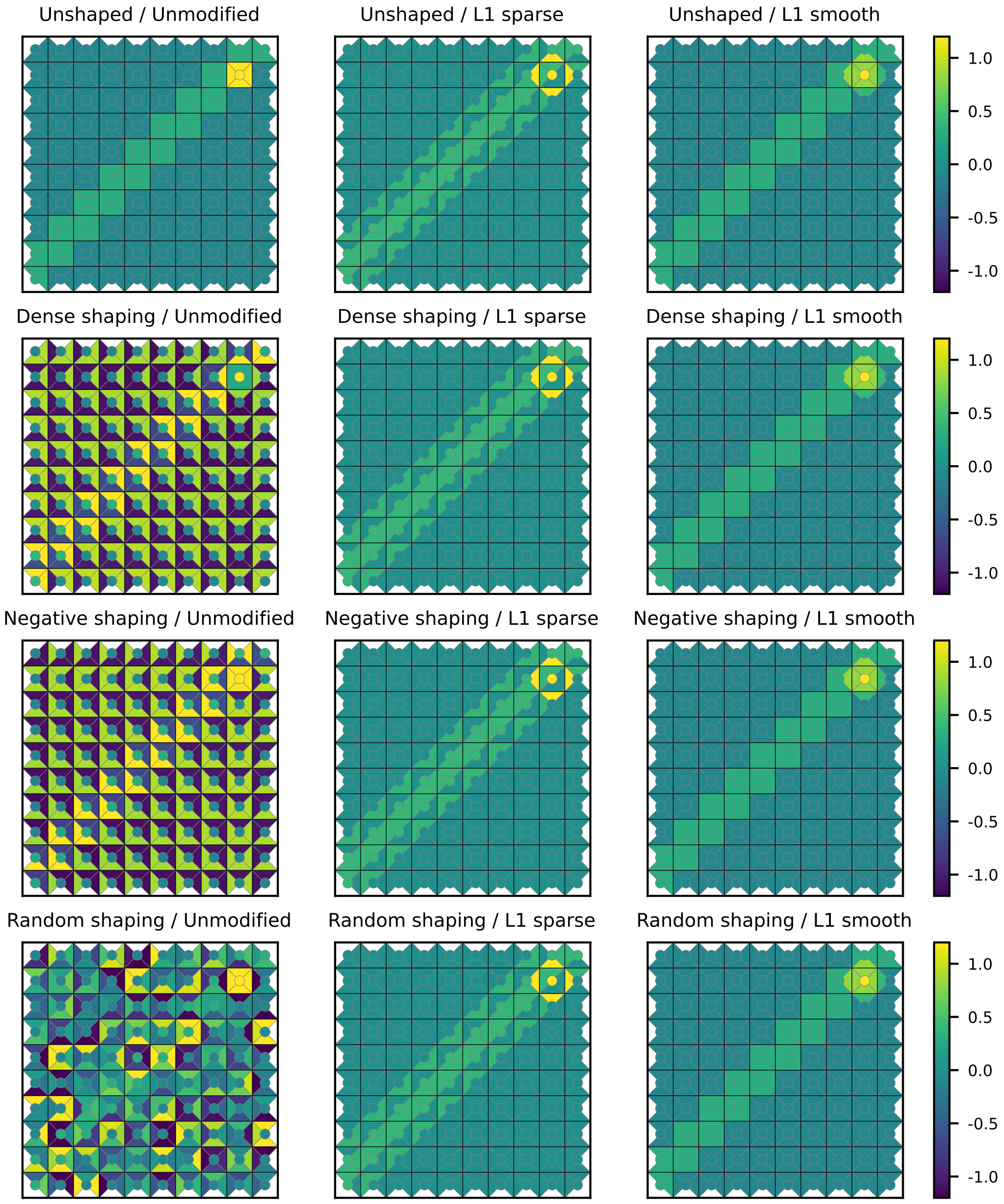


Figure C.3: `Path` reward preprocessed with L^1 versions of the sparsity and smoothness cost functions. The results are very similar to those in Figure 6.2. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

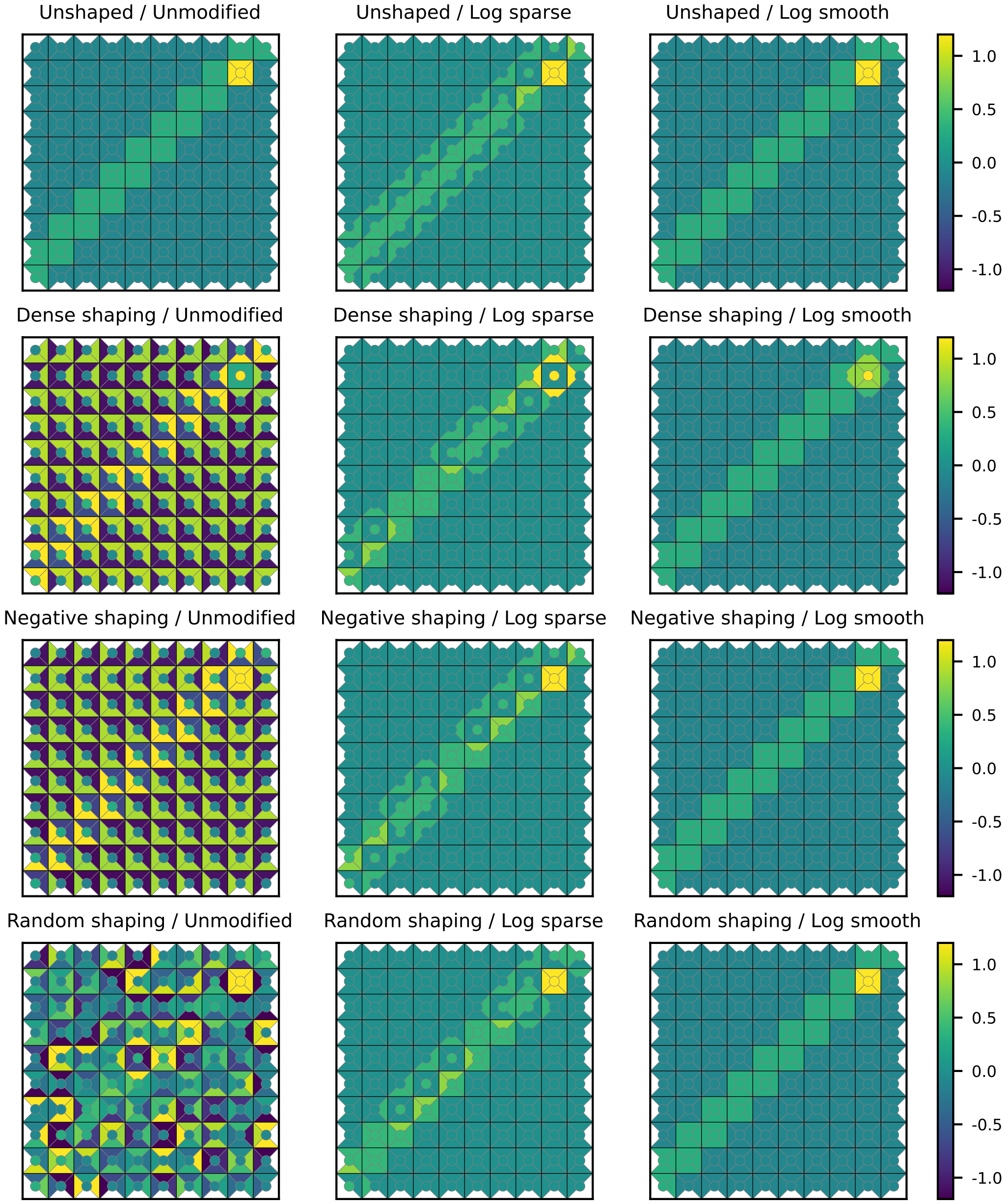


Figure C.4: `Path` reward preprocessed with logarithmic versions of the sparsity and smoothness cost functions. Compared to the L^1 sparse cost function in Figure C.3, the log sparse cost recovers a slightly less symmetric but still significantly simplified reward. Each heatmap shows the rewards for all possible transitions in a 10×10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

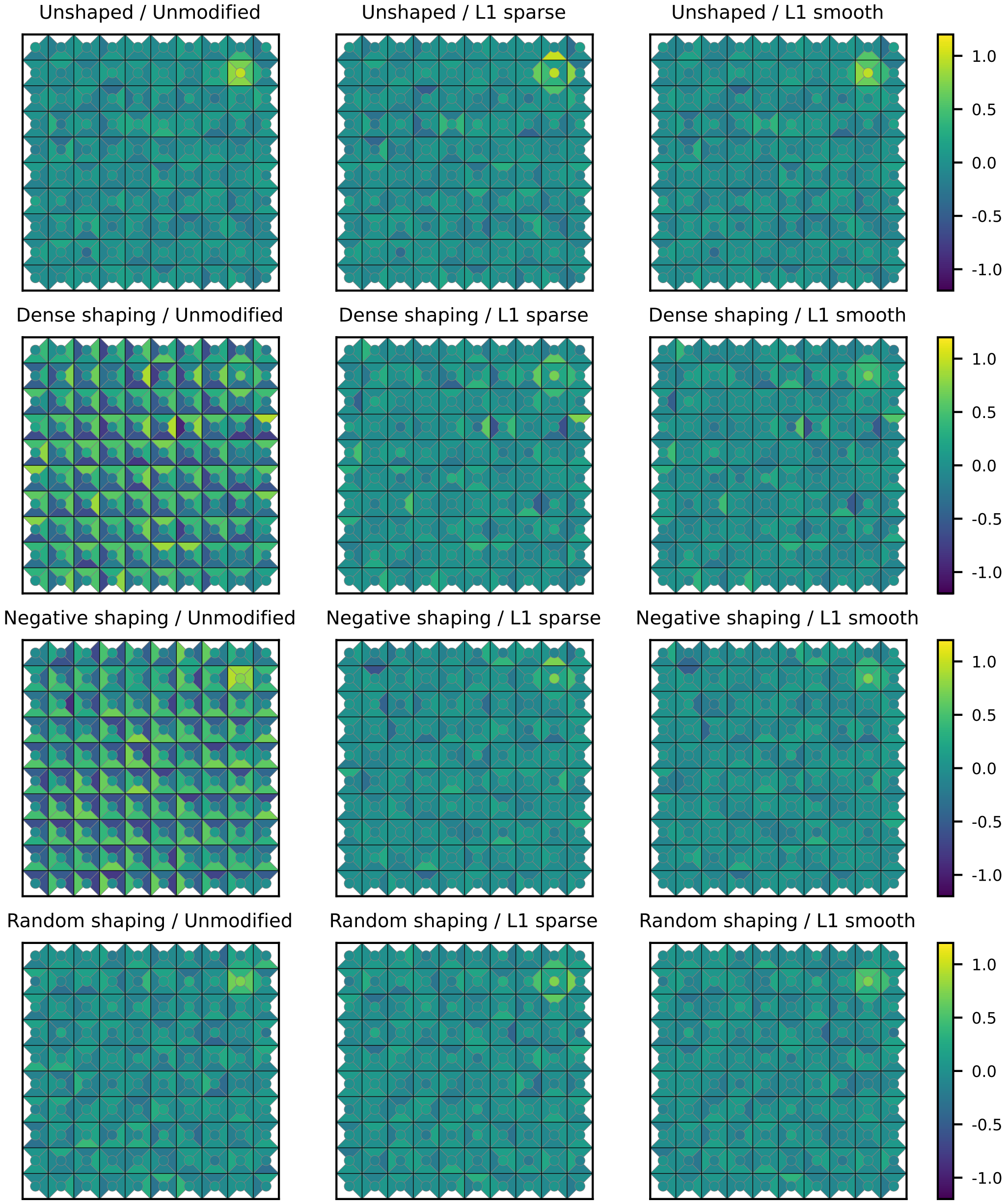


Figure C.5: Reward models trained on synthetic data from the `Goal` reward using preference comparison (leftmost column) and preprocessed versions of these (middle and right). The cost functions used here are the L^1 version of the sparsity and smoothness cost. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

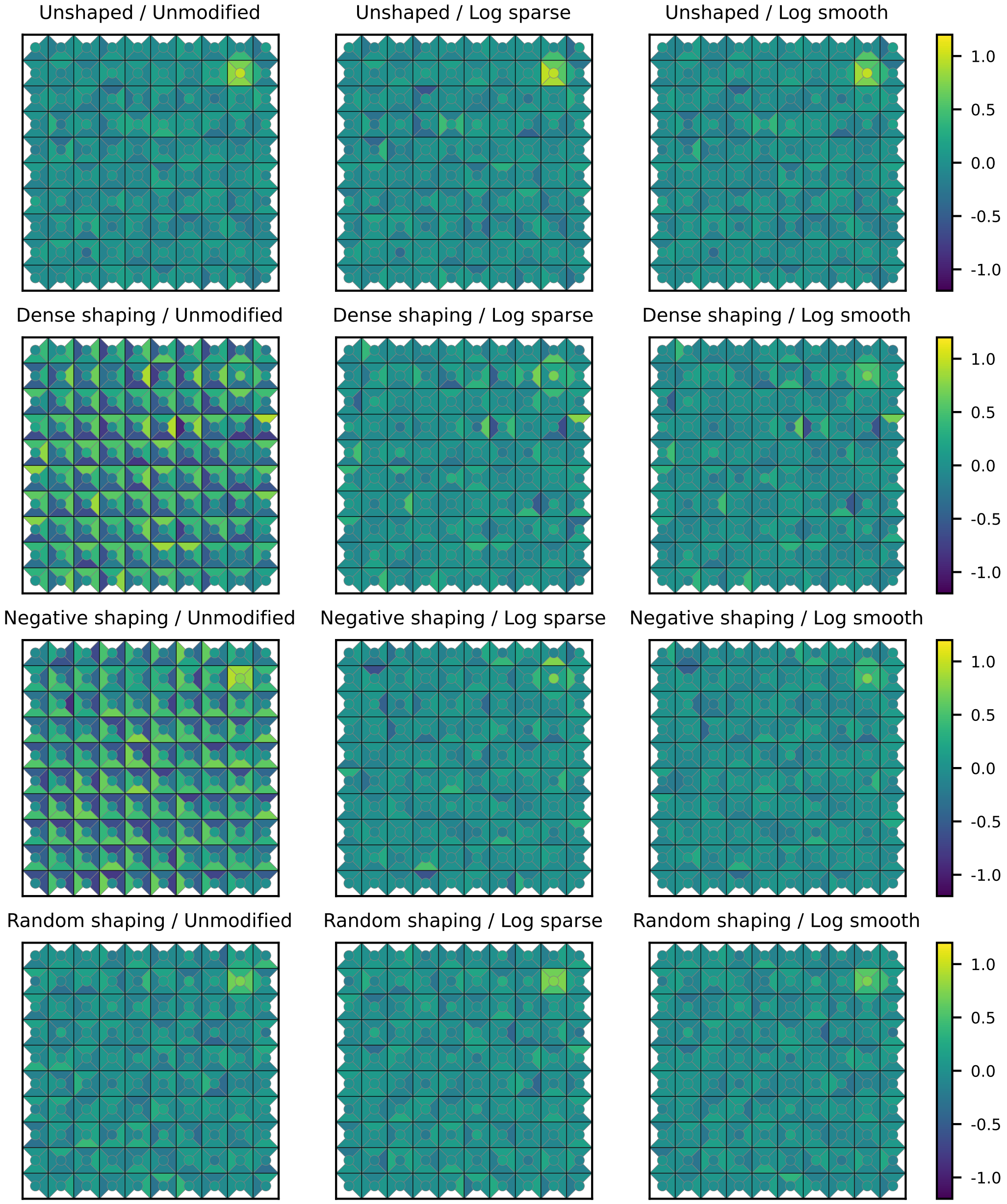


Figure C.6: Reward models trained on synthetic data from the `Goal` reward using preference comparison (leftmost column) and preprocessed versions of these (middle and right). The cost functions used here are the logarithmic version of the sparsity and smoothness cost. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

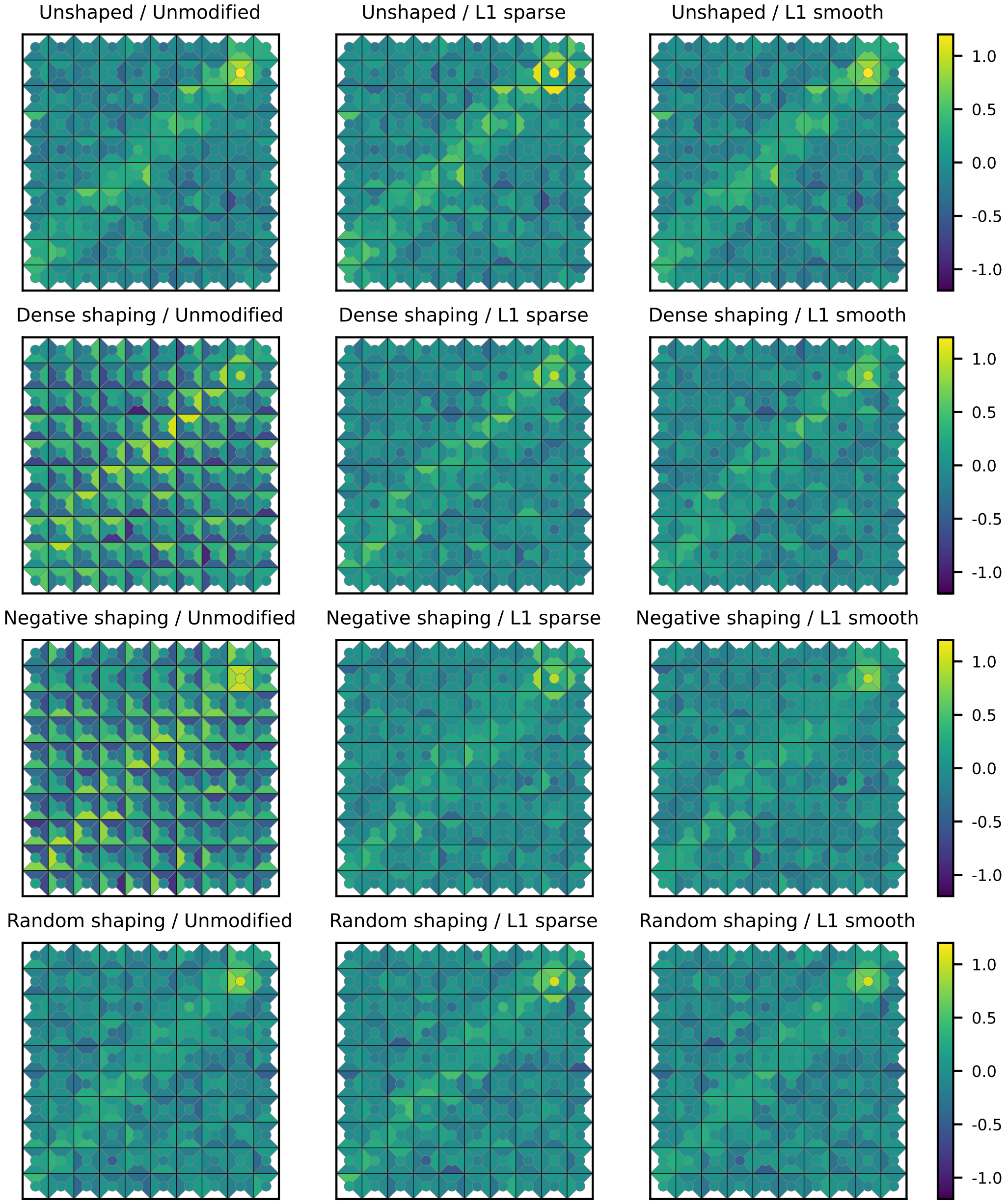


Figure C.7: Reward models trained on synthetic data from the `Path` reward using preference comparison (leftmost column) and preprocessed versions of these (middle and right). The cost functions used here are the L^1 version of the sparsity and smoothness cost. Each heatmap shows the rewards for all possible transitions in a 10×10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

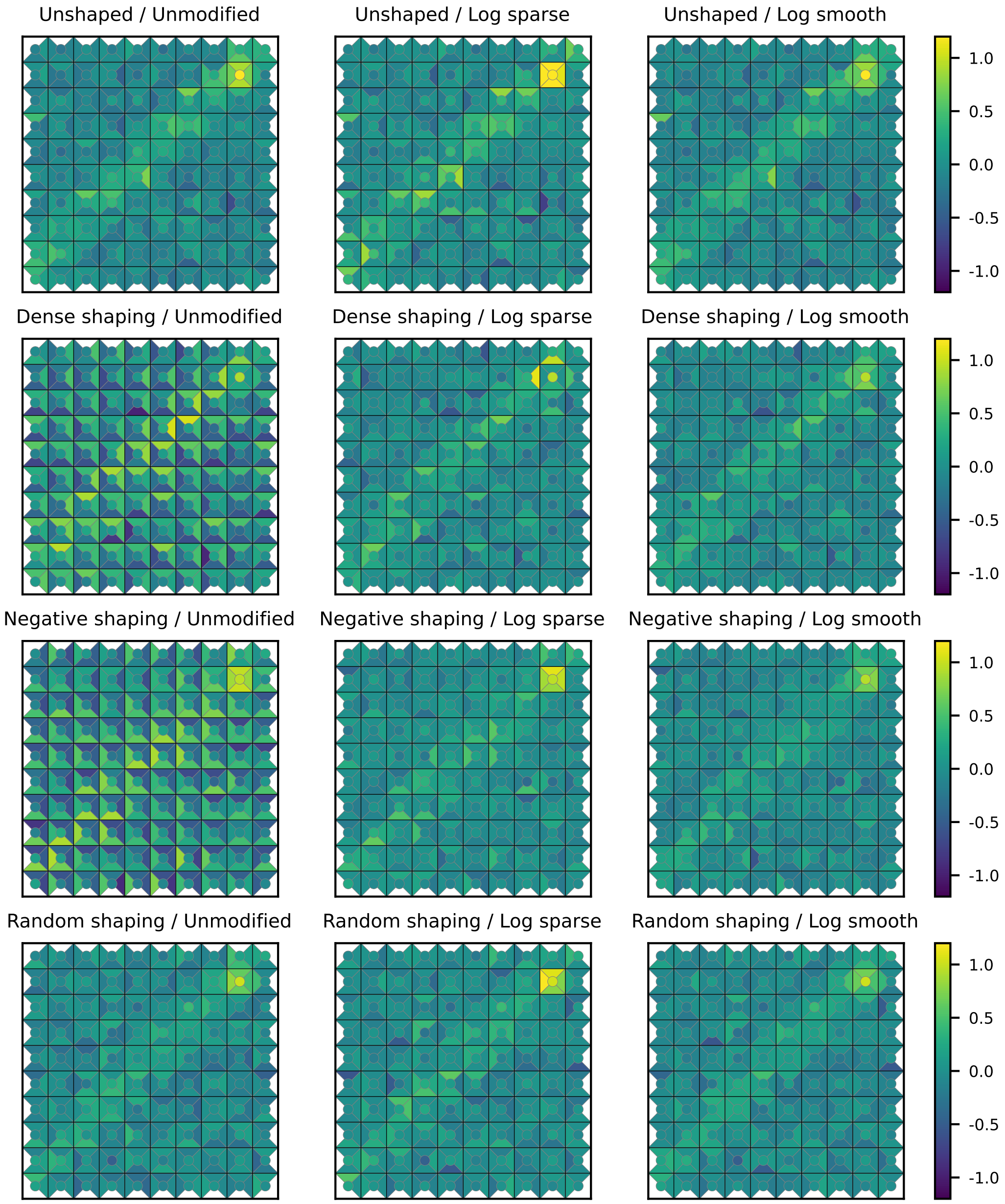


Figure C.8: Reward models trained on synthetic data from the `Path` reward using preference comparison (leftmost column) and preprocessed versions of these (middle and right). The cost functions used here are the logarithmic version of the sparsity and smoothness cost. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

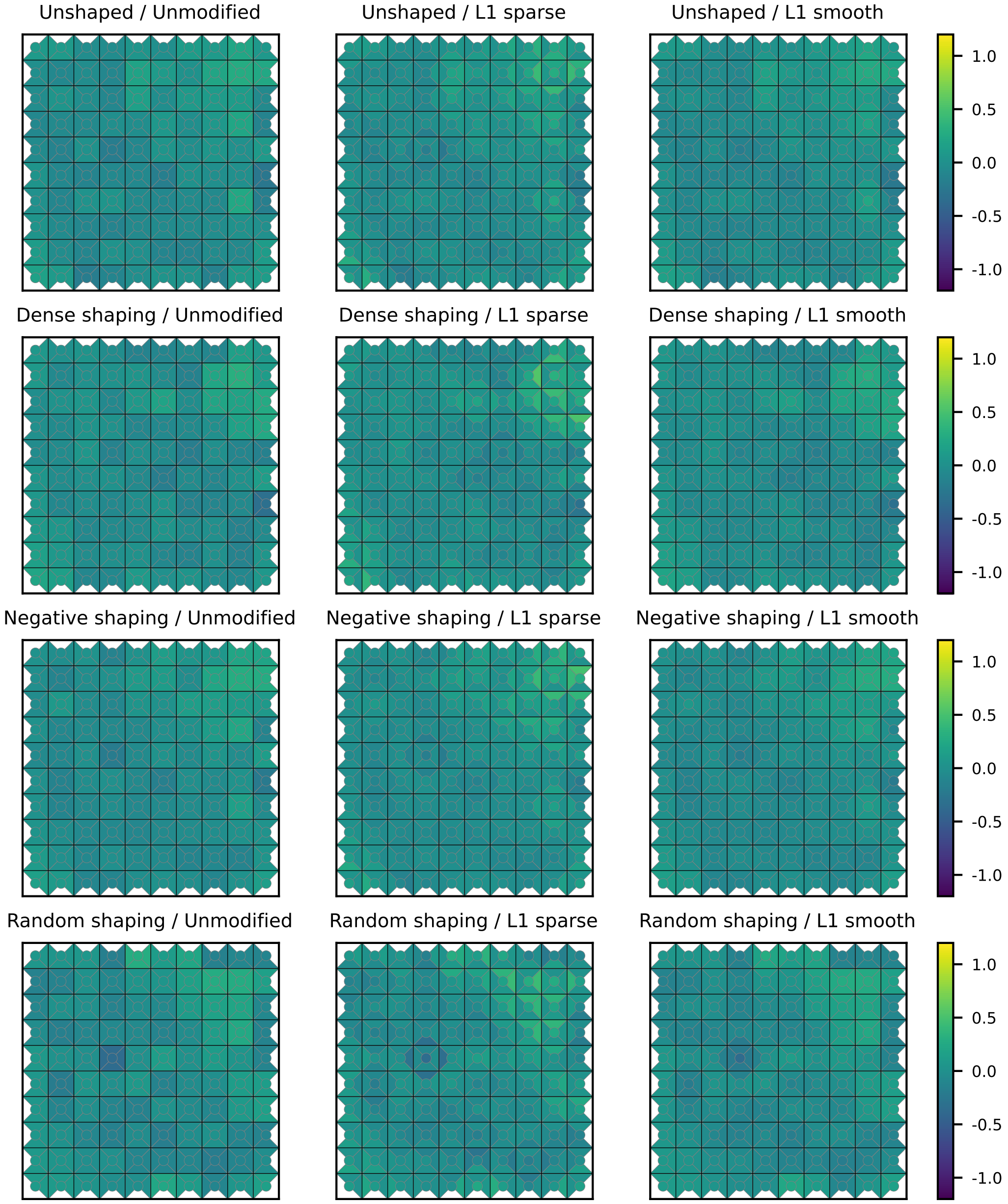


Figure C.9: Reward models learned using AIRL from expert demonstrations for the `Goal` reward (leftmost column) and preprocessed versions of these (middle and right). The cost functions used here are the L^1 versions of the sparsity and smoothness cost. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

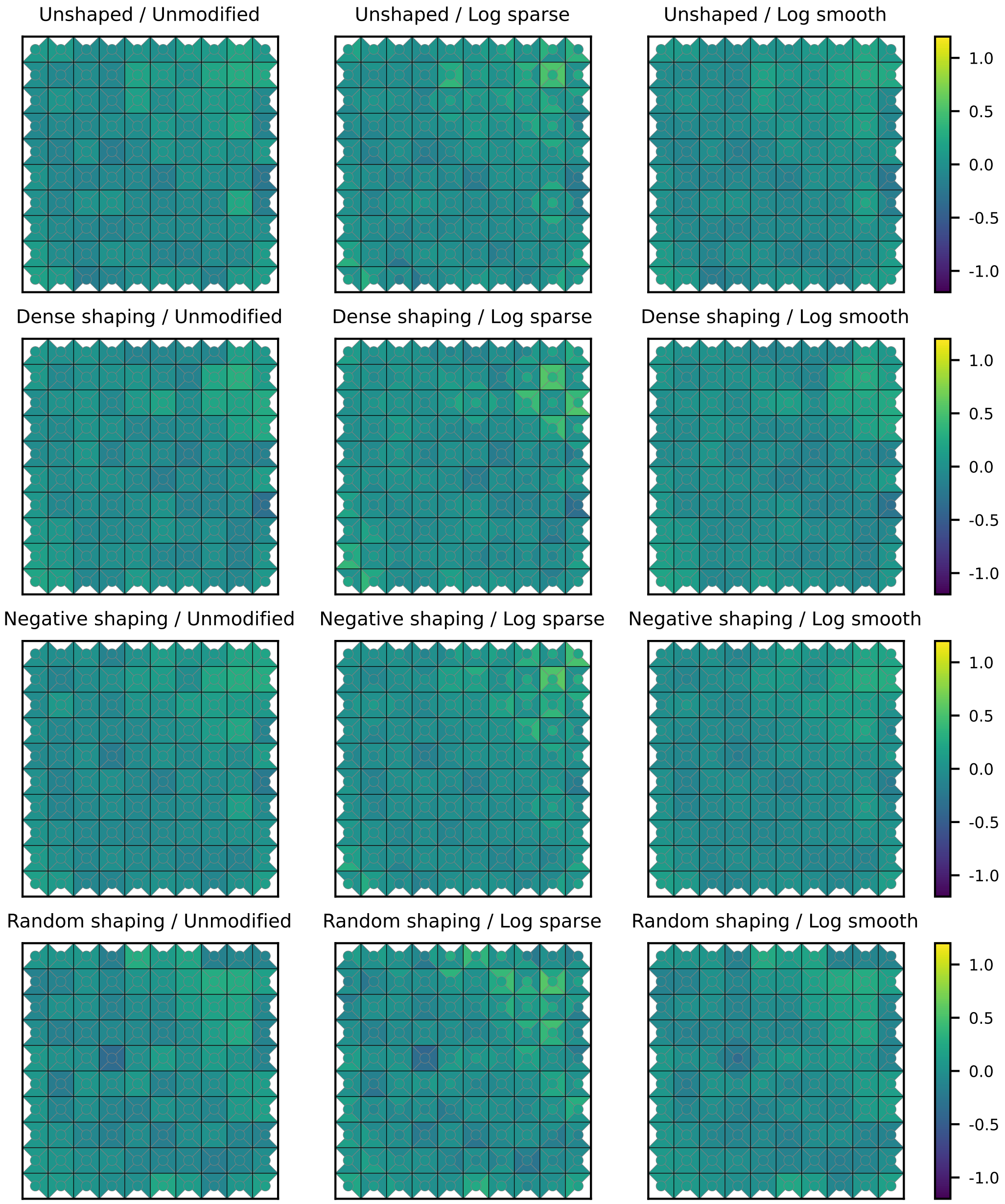


Figure C.10: Reward models learned using AIRL from expert demonstrations for the `Goal` reward (leftmost column) and preprocessed versions of these (middle and right). The cost functions used here are the logarithmic versions of the sparsity and smoothness cost. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

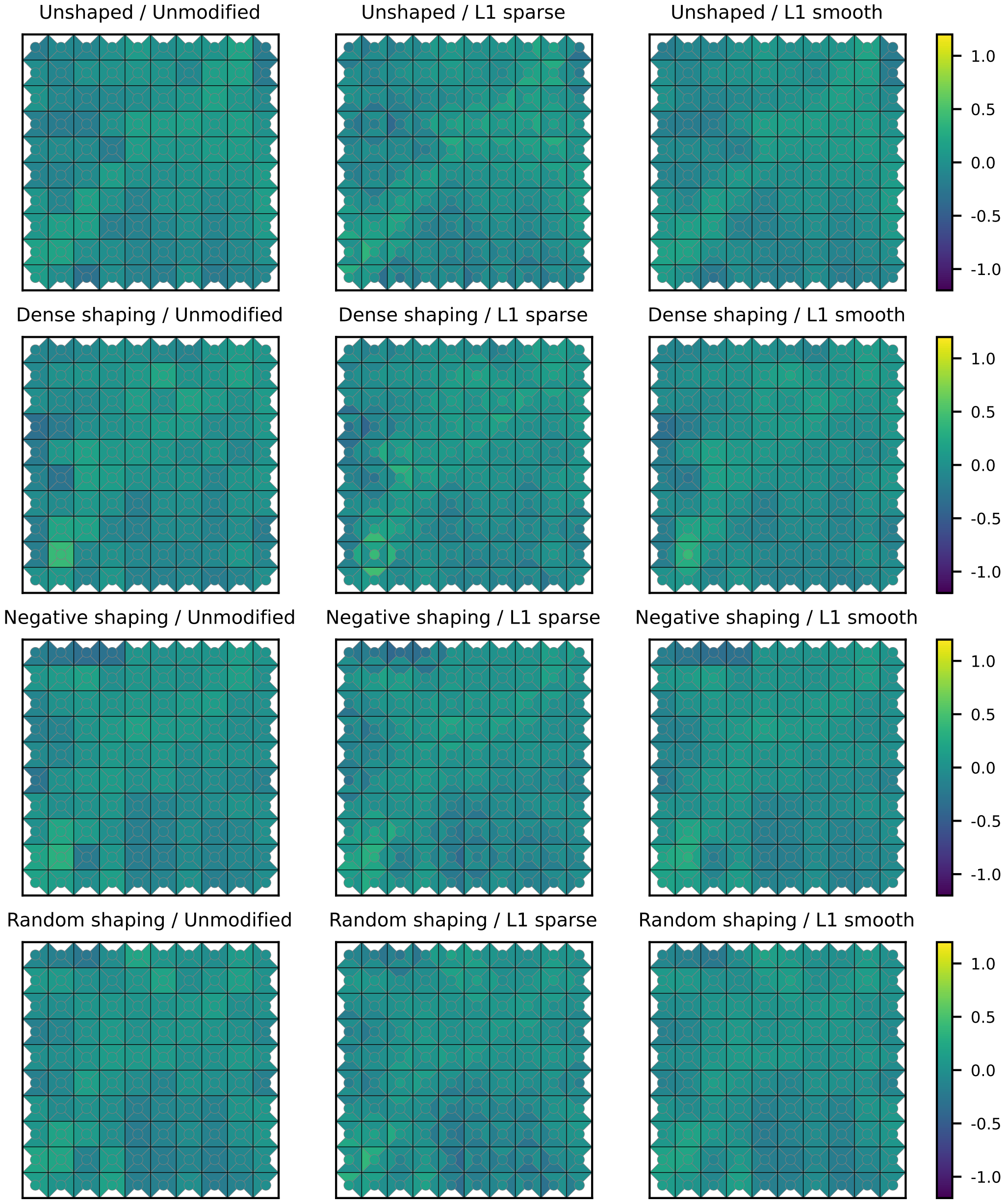


Figure C.11: Reward models learned using AIRL from expert demonstrations for the `Path` reward (leftmost column) and preprocessed versions of these (middle and right). The cost functions used here are the L^1 versions of the sparsity and smoothness cost. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

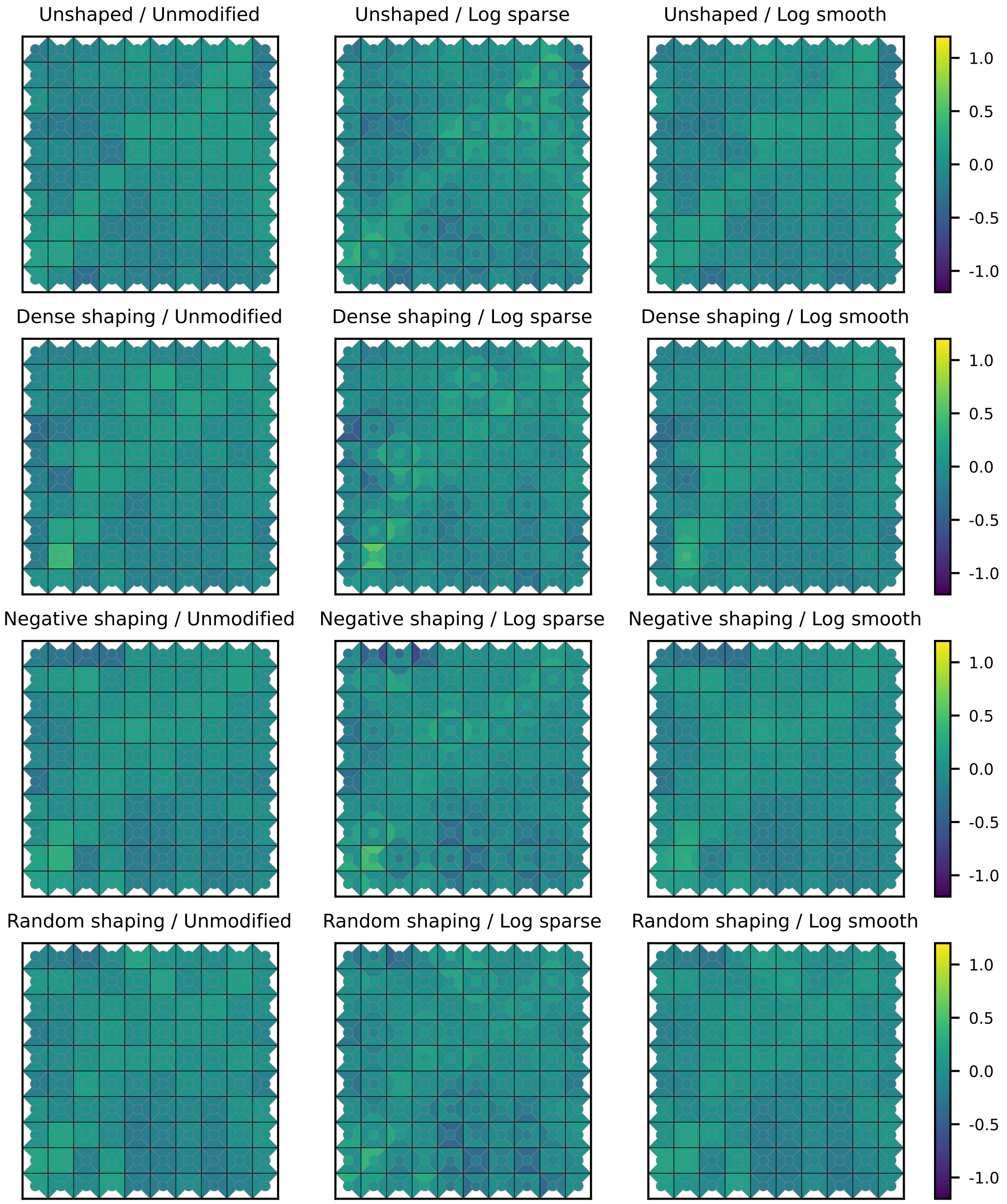


Figure C.12: Reward models learned using AIRL from expert demonstrations for the `Path` reward (leftmost column) and preprocessed versions of these (middle and right). The cost functions used here are the logarithmic versions of the sparsity and smoothness cost. Each heatmap shows the rewards for all possible transitions in a 10 × 10 gridworld. The circle in the center of each square represents the reward for staying in that state. The four triangles in each square represent the reward of transitions *leaving* that square in each of the four directions.

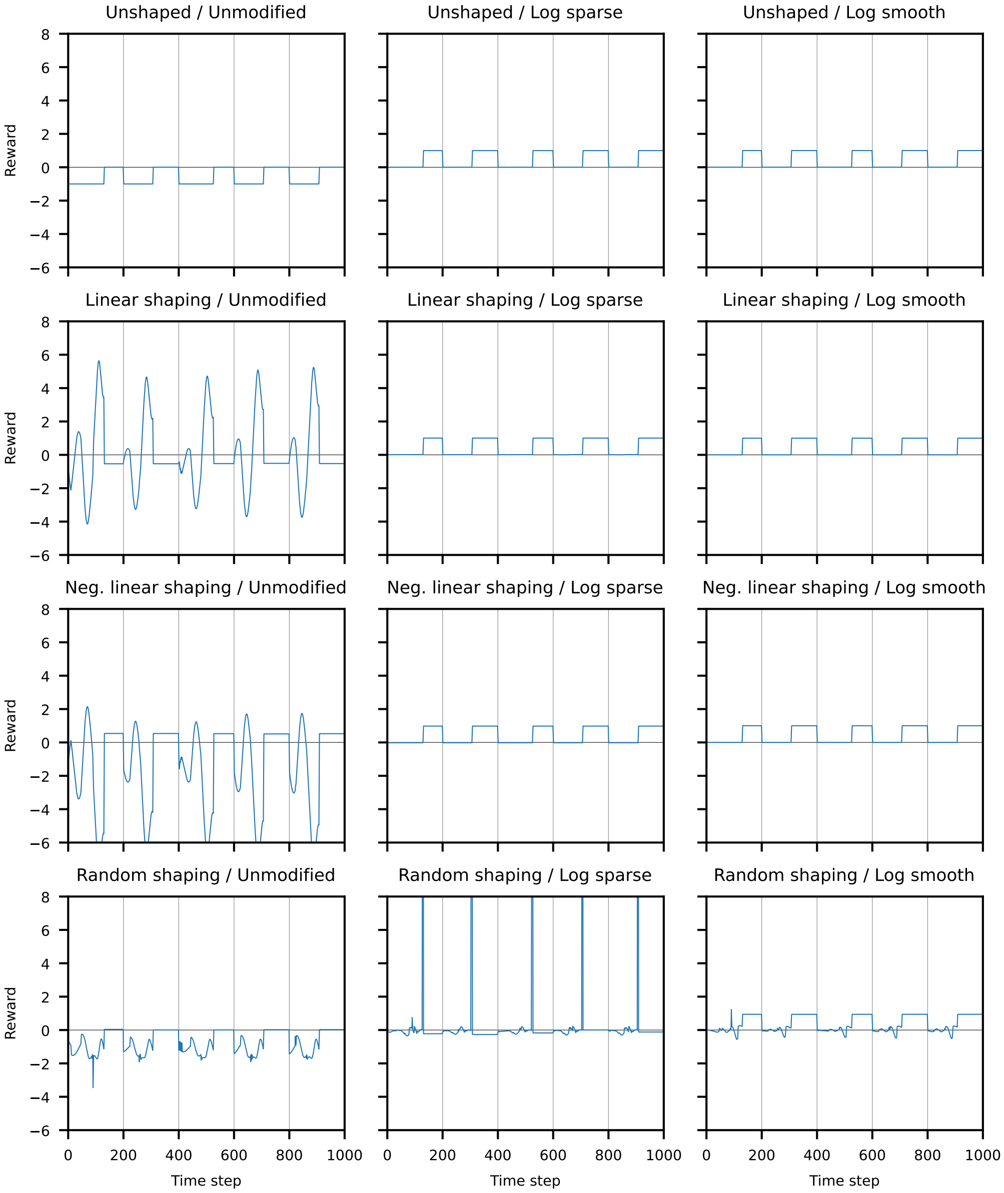


Figure C.13: Preprocessing simplifies rewards in the continuous mountain car environment. The top-left shows the ground-truth reward over time, with three shaped versions below. The middle and right columns show these rewards after preprocessing using the logarithmic sparsity and smoothness metrics. For the first two (linear) shapings, preprocessing recovers the ground truth reward exactly (up to a constant shift). In the more complex case in the last row, preprocessing still significantly simplifies the reward. See Figure C.14 for versions with an L^1 cost function. Each plot shows the reward during a rollout over five episodes (separated by the gray vertical lines).

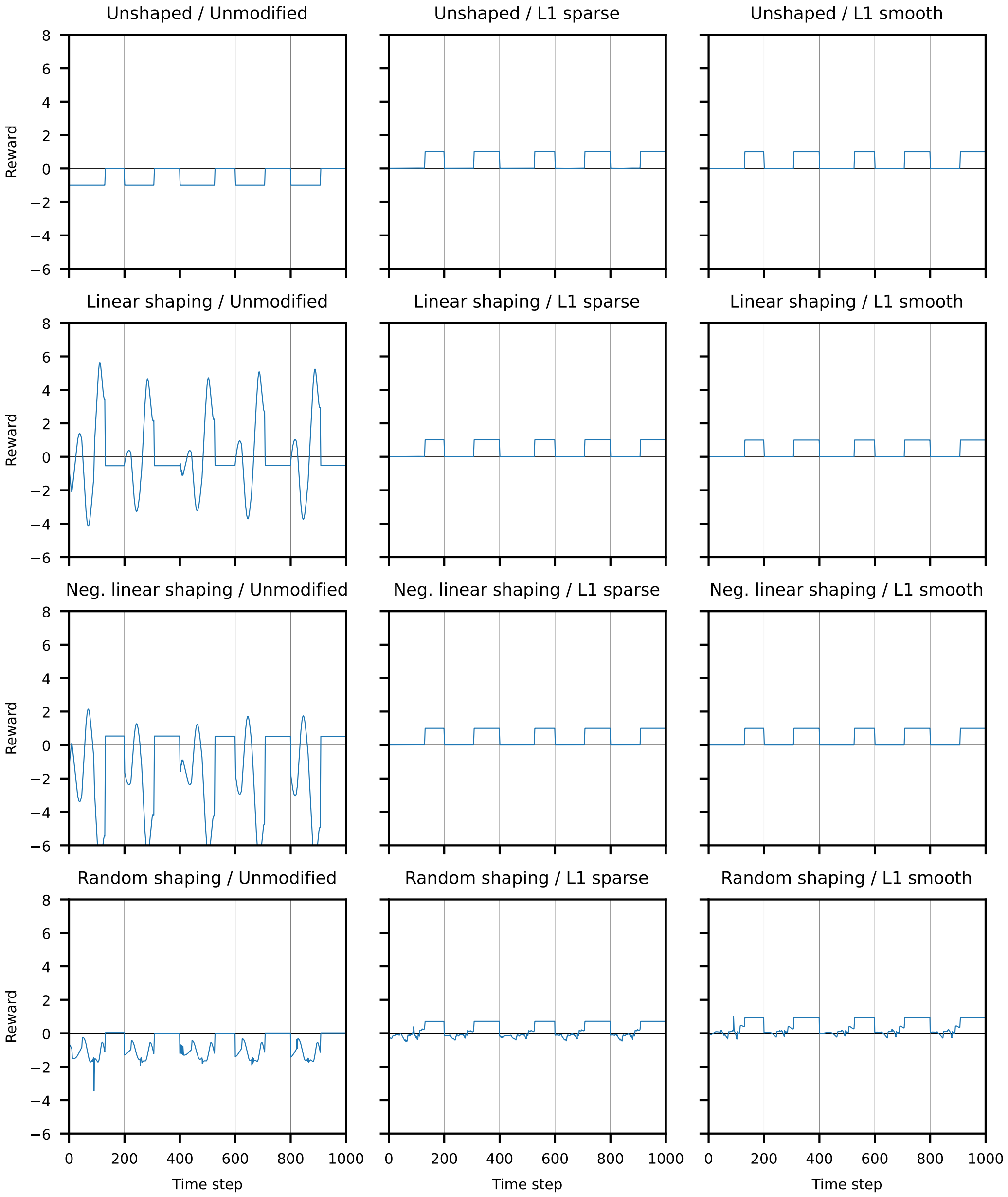


Figure C.14: The top-left shows the ground-truth reward in mountain car over time, with three shaped versions below. The middle and right column show these rewards after preprocessing using the L^1 sparsity and smoothness metrics. This works reasonably well for these simple shaped rewards, although in the more complex last row these cost functions appear to perform less well than the logarithmic version in Figure C.13. Each plot shows the reward during a rollout over five episodes (separated by the gray vertical lines).

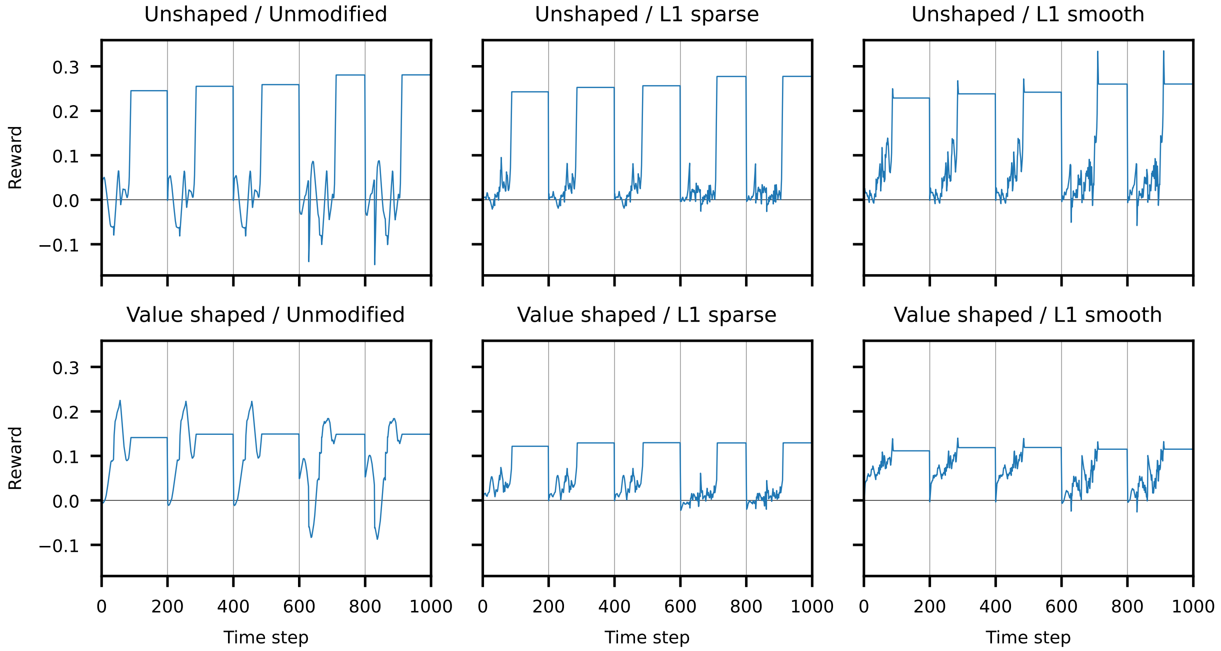


Figure C.15: Preprocessing can simplify complex learned reward models for mountain car. The left column shows reward models learned using synthetic preference comparisons based on the ground-truth reward (top), and the ground-truth shaped with an optimal value function (bottom). Preprocessing for sparsity (middle) and smoothness (right) produces simpler and less noisy reward curves, especially in the shaped setting. The results are extremely similar, although perhaps slightly worse, than the logarithmic version used in Figure 6.3. Each plot shows the reward during a rollout over five episodes (separated by the gray vertical lines).

Appendix D

Deferred content from Chapter 8

D.1 Training: hyperparameters and computational infrastructure

Parameter	Value	Search Range	Search Distribution
Total Timesteps	20×10^6	$[0, 40 \times 10^6]$	Manual
Batch size	16 384	$[2048, 65\,536]$	Log uniform
Number of environments	8	$[1, 16]$	Manual
Mini-batches	4	$[1, 128]$	Log uniform
Epochs per update	4	$[1, 11]$	Uniform
Learning rate	3×10^{-4}	$[1 \times 10^{-5}, 1 \times 10^{-2}]$	Log uniform
Discount	0.99	—	—
Maximum Gradient Norm	0.5	—	—
Clip Range	0.2	—	—
Advantage Estimation Discount	0.95	—	—
Entropy coefficient	0.0	—	—
Value Function Loss Coefficient	0.5	—	—

Table D.1: Hyperparameters for Proximal Policy Optimization.

Table D.1 specifies the hyperparameters used for training. The number of environments was chosen for performance reasons after observing diminishing returns from using more than 8 parallel environments. The total timesteps was chosen by inspection after observing diminishing returns to additional training. The batch size, mini-batches, epochs per update, entropy coefficient and learning rate were tuned via a random search with 100 samples on two environments, *Kick and Defend* and *Sumo Humans*. All other hyperparameters are the defaults in the `PPO2` implementation in Stable Baselines [65].

We repeated the hyperparameter sweep for fine-tuning defender policies for the defense experiments, but obtained similar results. For simplicity, we therefore chose to use the same hyperparameters throughout.

We used a mixture of in-house and cloud infrastructure to perform these experiments. It takes around 8 hours to train an adversary for a single defender using 4 cores of an Intel Xeon Platinum 8000 (Skylake) processor.

D.2 Activation analysis: t-SNE and GMM

We collect activations from all feed forward layers of the defender's policy network. This gives two 64-length vectors, which we concatenate into a single 128-dimension vector for analysis with a Gaussian Mixture Model and a t-SNE representation.

D.2.1 t-SNE hyperparameter selection

We fit models with perplexity 5, 10, 20, 50, 75, 100, 250 and 1000. We chose 250 since qualitatively it produced the clearest visualization of data with a moderate number of distinct clusters.

D.2.2 Gaussian Mixture Model hyperparameter selection

We fit models with 5, 10, 20, 40 and 80 components with a full (unrestricted) and diagonal covariance matrix. We used the Bayesian Information Criterion (BIC) and average log-likelihood on a held-out validation set as criteria for selecting hyperparameters. We found 20 components with a full covariance matrix achieved the lowest BIC and highest validation log-likelihood in the majority of environment-defender pairs, and was the runner-up in the remainder.

D.3 Figures

Supplementary figures are provided on the subsequent pages.

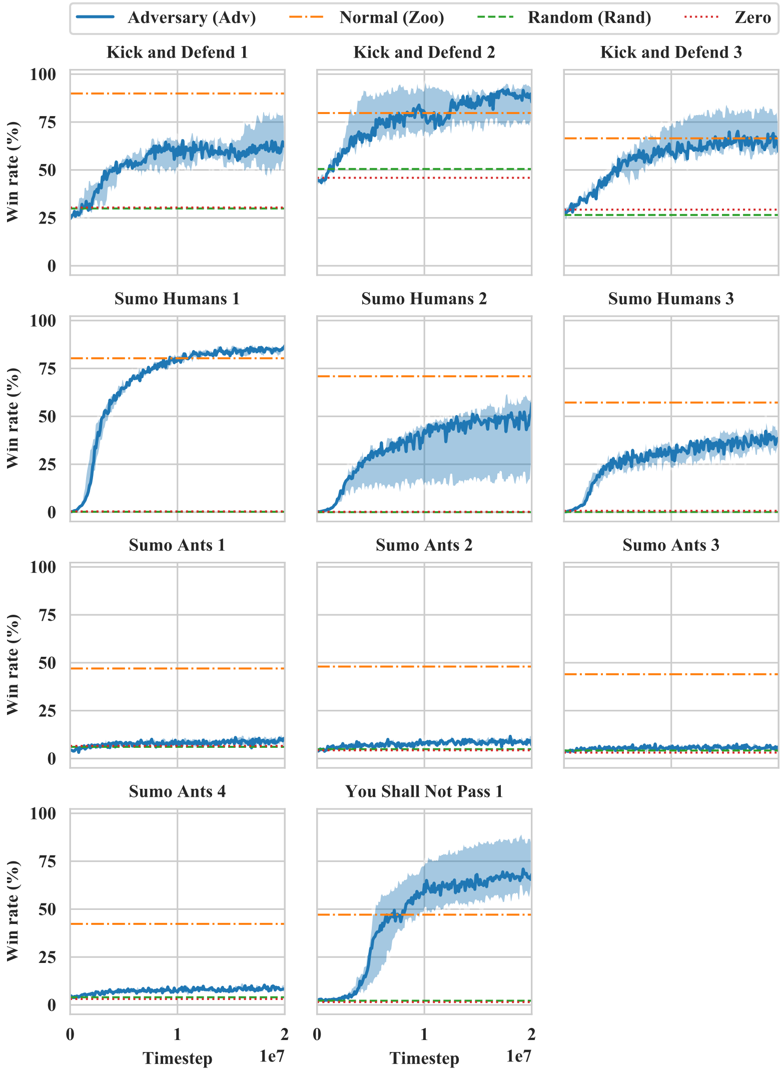
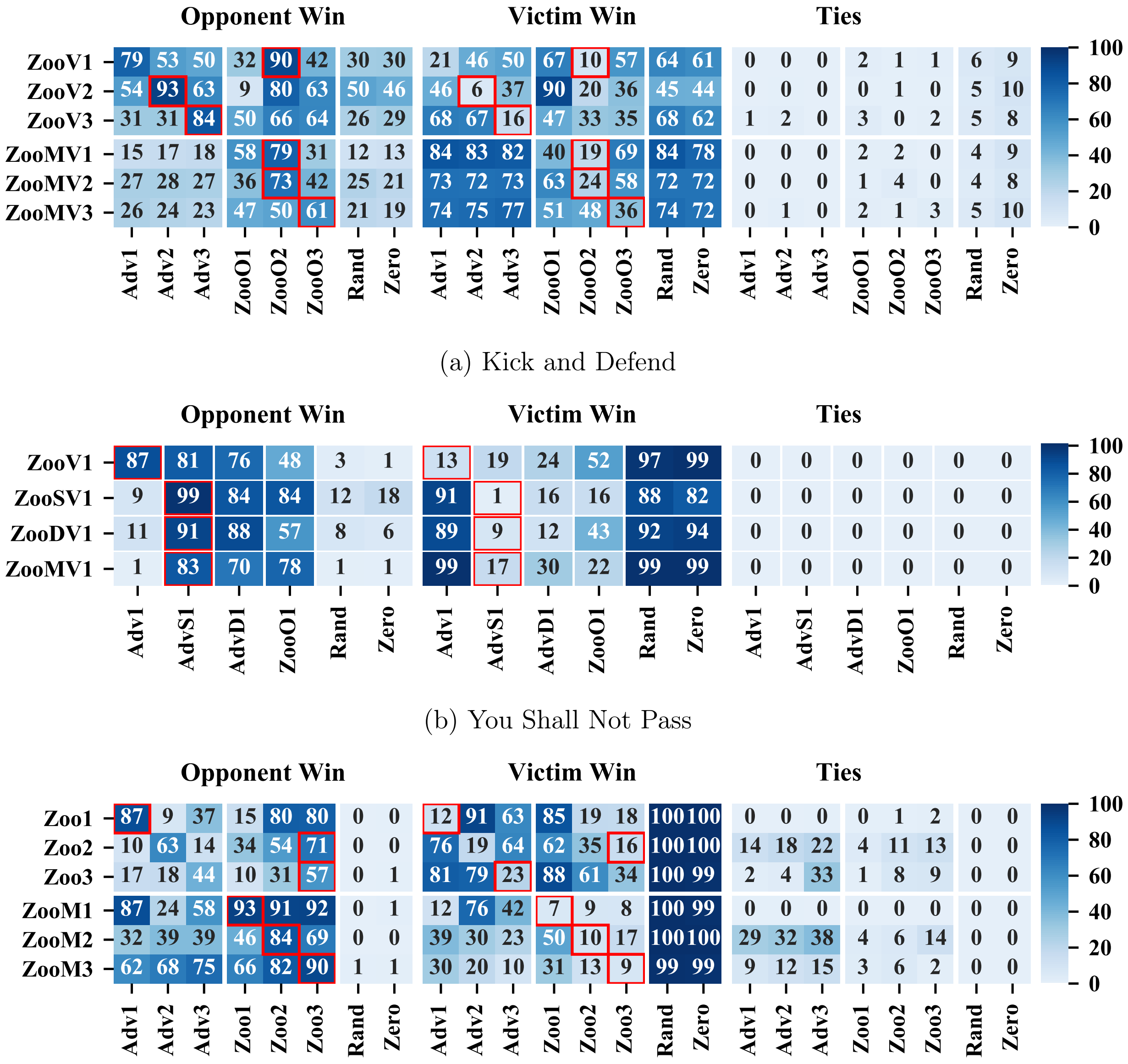


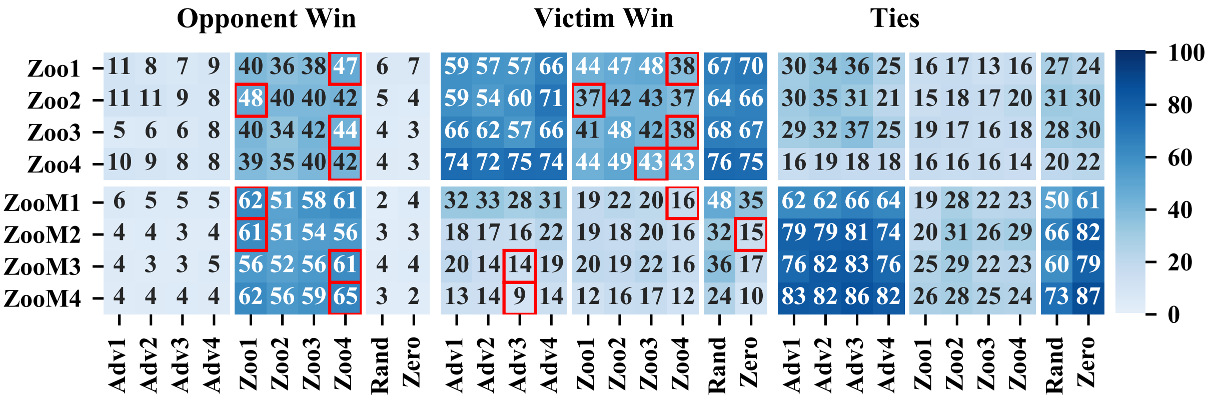
Figure D.1: Win rates while training adversary `Adv`. The adversary exceeds baseline win rates against most defenders in *Kick and Defend* and *You Shall Not Pass*, is competitive on *Sumo Humans*, but performs poorly in the low-dimensional *Sumo Ants* environment. **Key**: The solid line shows the median win rate for `Adv` across 5 random seeds, with the shaded region representing the minimum and maximum. The win rate is smoothed with a rolling average over 100 000 timesteps. Baselines are shown as horizontal dashed lines. Agents `Rand` and `Zero` take random and zero actions respectively. The `Zoo` baseline is whichever `ZooM` (*Sumo*) or `ZooOM` (other environments) agent achieves the highest win rate. The defender is `ZooN` (*Sumo*) or `ZooVN` (other environments), where N is given in the title above each figure.



(a) Kick and Defend

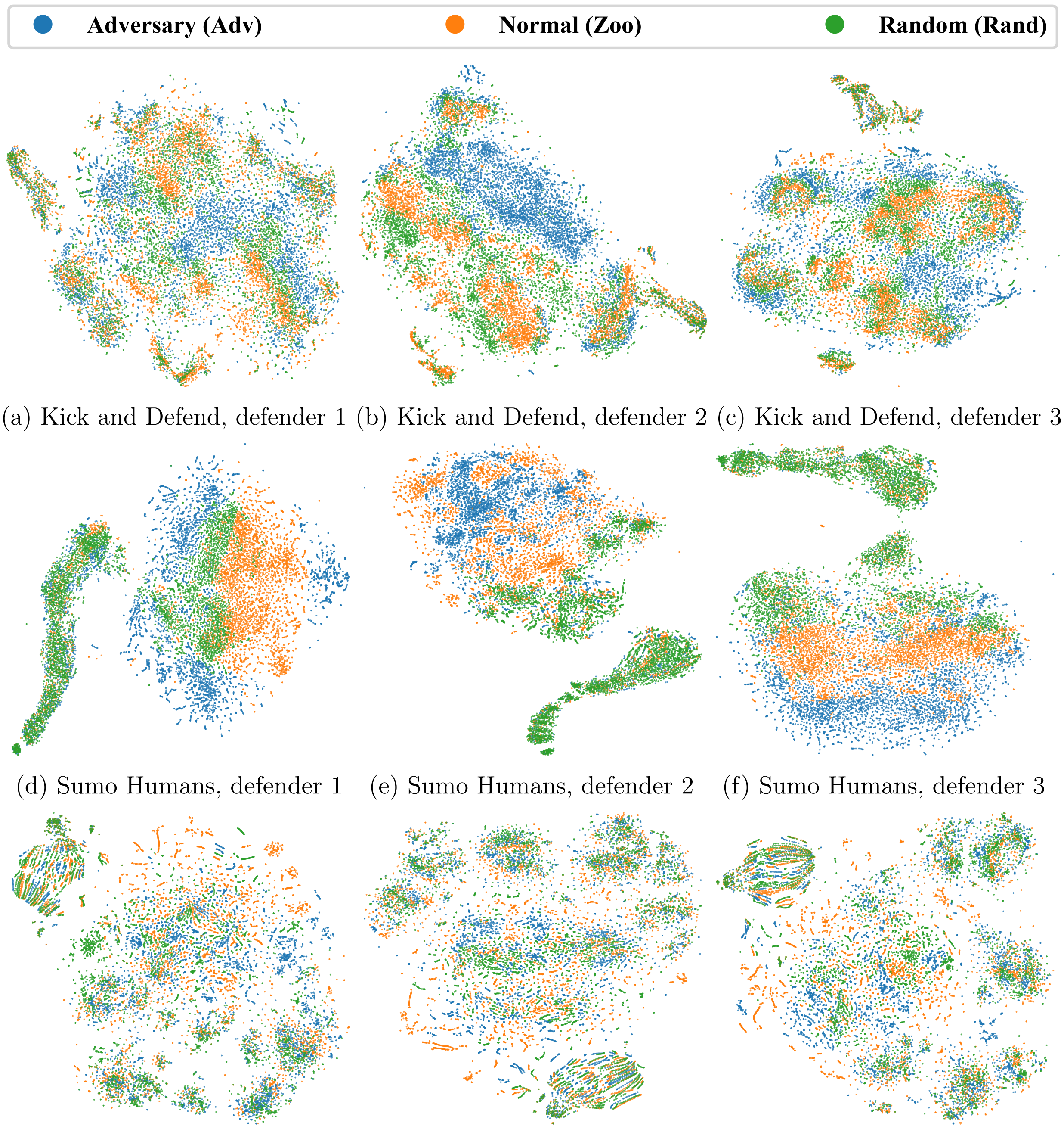
(b) You Shall Not Pass

(c) Sumo Humans



(d) Sumo Ants

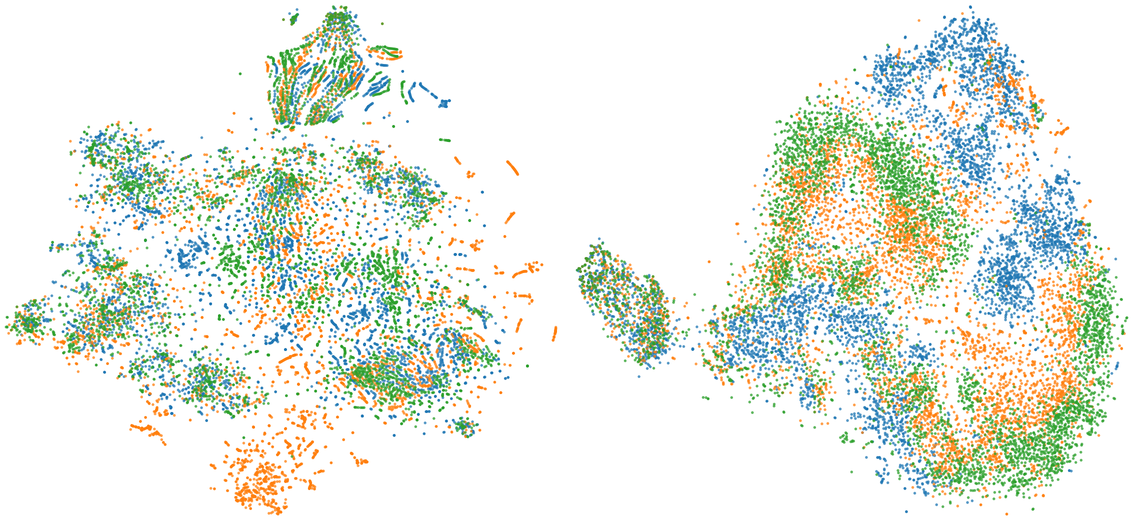
Figure D.2: Percentage of episodes (out of 1000) won by the opponent, the defender or tied. The maximal opponent win rate in each row is in red. Defenders are on the y-axis and opponents on the x-axis. **Key**: Agents `ZooYN` are pre-trained policies from Bansal et al. [14], where $Y \in \{‘V’,‘O’,‘’\}$ denotes the agent plays as **(V)**ictim, **(O)**pponent, or either side, and N is a random seed. Opponents `AdvN` are the best adversarial policy of 5 seeds trained against the corresponding `Zoo[V]N`. Agents `Rand` and `Zero` are baseline agents taking random and zero actions respectively. Hardened defenders `ZooXYN`, where $X \in \{‘S’,‘D’,‘M’\}$, are derived from `ZooYN` by fine-tuning against a **(S)**ingle opponent `AdvN`, or **(D)**ual opponents `AdvN` and `Zoo[O]N`, or by **(M)**asking the observations.



(a) Kick and Defend, defender 1 (b) Kick and Defend, defender 2 (c) Kick and Defend, defender 3

(d) Sumo Humans, defender 1 (e) Sumo Humans, defender 2 (f) Sumo Humans, defender 3

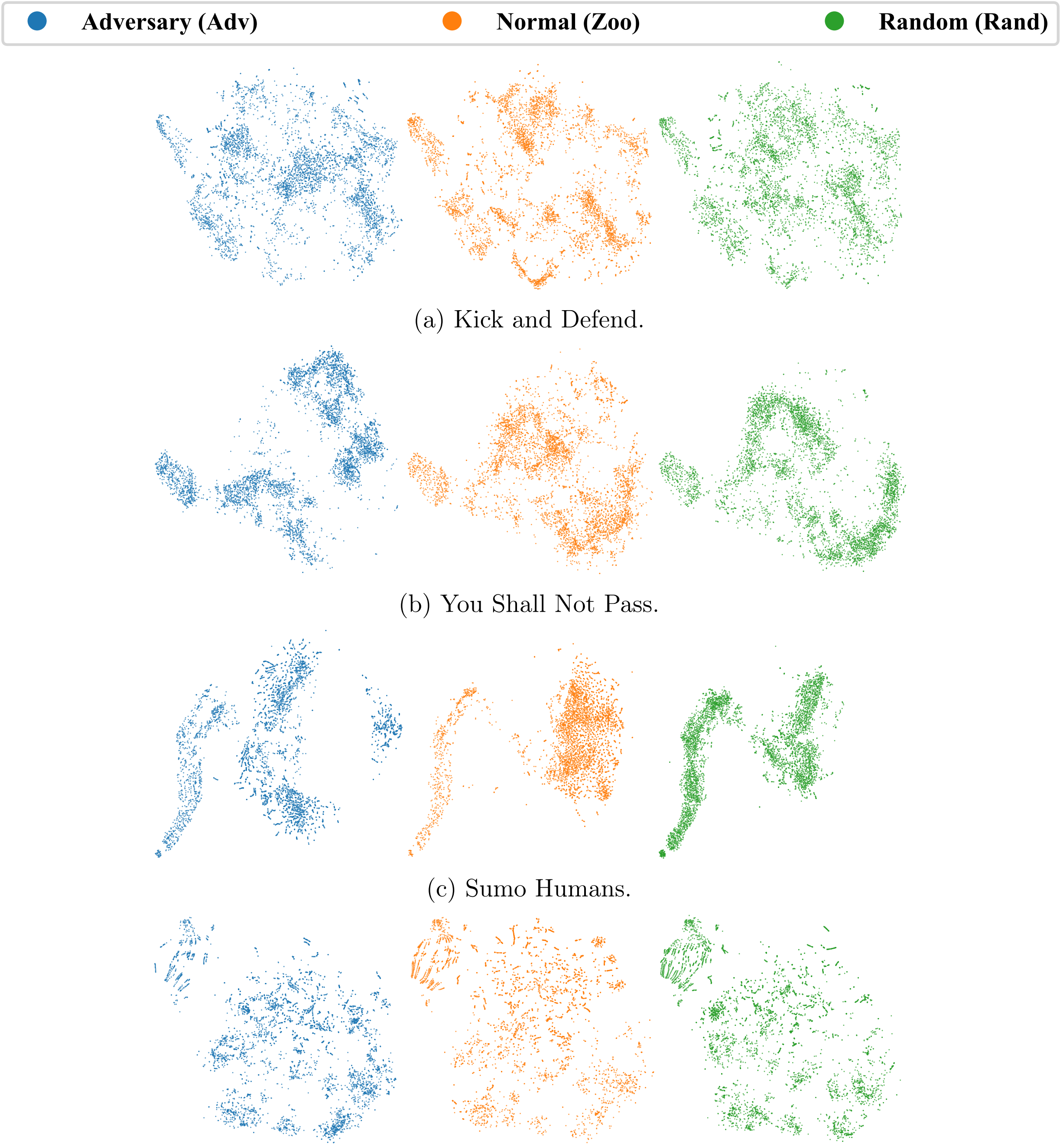
(g) Sumo Ants, defender 1 (h) Sumo Ants, defender 2 (i) Sumo Ants, defender 3



(j) Sumo Ants, defender 4

(k) You Shall Not Pass, defender 1

Figure D.3: t-SNE activations of the defender when playing against different opponents. There is a clear separation between the activations induced by `Adv` and those of the normal opponent `Zoo`. Model fitted with a perplexity of 250 to activations from 5000 timesteps against each opponent. The defender is `ZooN` (*Sumo*) or `ZooVN` (other environments), where N is given in the figure caption. Opponent `Adv` is the best adversary trained against the defender. Opponent `Zoo` corresponds to `ZooN` (*Sumo*) or `ZooON` (other environments). See Figure D.4 for activations for a single opponent at a time.



(a) Kick and Defend.

(b) You Shall Not Pass.

(c) Sumo Humans.

(d) Sumo Ants.

Figure D.4: t-SNE activations of defender `Zoo1` (*Sumo*) or `ZooV1` (other environments). The results are the same as in Figure D.3 but decomposed into individual opponents for clarity. Model fitted with a perplexity of 250 to activations from 5000 timesteps against each opponent. Opponent `Adv` is the best adversary trained against the defender. Opponent `Zoo` is `Zoo1` (*Sumo*) or `ZooO1` (other environments). See Figure D.3 for results for other defenders (one plot per defender).

Appendix E

Deferred content from Chapter 9

E.1 Rules of Go Used For Evaluation

We evaluate all games with Tromp-Taylor rules [163], after clearing opposite-color stones within pass-alive groups computed by Benson's algorithm [20]. KataGo was configured to play using these rules in all our matches against it. Indeed, these rules simply consist of KataGo's version of Tromp-Taylor rules with `SelfPlayOpts` enabled [177]. We use a fixed Komi of 6.5.

We chose these *modified Tromp-Taylor* rules because they are simple, and KataGo was trained on (variants) of these rules so should be strongest playing with them. Although the exact rules used were randomized during KataGo's training, modified Tromp-Taylor made up a plurality of the training data. That is, modified Tromp-Taylor is at least as likely as any other configuration seen during training, and is more common than some other options.[*]

In particular, KataGo training randomized between area vs. territory scoring as well as ko, suicide, taxation and button rules from the options described in Wu [177]. These configuration settings are provided as input to the neural network [175, Table 4], so the network should learn to play appropriately under a range of rule sets. Additionally, during training Komi was sampled randomly from a normal distribution with mean 7 and standard deviation 1 [175, Appendix D].

E.1.1 Difference From Typical Human Play

Although KataGo supports a variety of rules, all of them involve automatically scoring the board at the end of the game. By contrast, when a match between humans end, the players typically confer and agree which stones are dead, removing them from the board prior to

[*] In private communication, the author of KataGo estimated that modified Tromp-Taylor made up a "a few %" of the training data, "growing to more like 10% or as much as 20%" depending on differences such as "self-capture and ko rules that shouldn't matter for what you're investigating, but aren't fully the same rules as Tromp-Taylor".

scoring. If no agreement can be reached then the players either continue playing the game until the situation is clarified, or a referee arbitrates the outcome of the game.

KataGo has a variety of optional features to help it play well under human scoring rules. For example, KataGo includes an auxiliary prediction head for whether stones are dead or alive. This enables it to propose which stones it believes are dead when playing on online Go servers. Additionally, it include hard-coded features that can be enabled to make it play in a more human-like way, such as `friendlyPassOk` to promote passing when heuristics suggest the game is nearly over.

These features have led some to speculate that the defender passes prematurely in games such as those in Figure 9.1 because it has learned or is configured to play in a more human-like way. *Prima facie*, this view seems credible: a human player certainly might pass in a similar situation to our defender, viewing the game as already won under human rules. Although tempting, this explanation is not correct: the optional features described above were disabled in our evaluation. Therefore KataGo loses under the rules it was both trained and configured to use.

In fact, the majority of our evaluation used the `match` command to run KataGo vs. KataGo agents which naturally does not support these human-like game play features. We did use the `gtp` command, implementing the Go Text Protocol (GTP), for a minority of our experiments, such as evaluating KataGo against other AI systems or human players. In those experiments, we configured `gtp` to follow the same Tromp-Taylor rules described above, with any human-like extensions disabled.

E.2 Search Algorithms

E.2.1 A Review of Monte-Carlo Tree Search (MCTS)

In this section, we review the basic Monte-Carlo Tree Search (MCTS) algorithm as used in AlphaGo-style agents [151]. This formulation is heavily inspired by the description of MCTS given in Wu [175].

MCTS is an algorithm for growing a game tree one node at a time. It starts from a tree T_0 with a single root node x_0. It then goes through N *playouts*, where every playout adds a leaf node to the tree. We will use T_i to denote the game tree after i playouts, and will use x_i to denote the node that was added to T_{i-1} to get T_i. After MCTS finishes, we have a tree T_N with $N + 1$ nodes. We then use simple statistics of T_N to derive a sampling distribution for the next move.

E.2.1.1 MCTS Playouts

MCTS playouts are governed by two learned functions:

a. A value function estimator $\hat{V} : \mathcal{T} \times \mathcal{X} \to \mathbb{R}$, which returns a real number $\hat{V}_T(x)$ given a tree T and a node x in T. The value function estimator is meant to estimate how good it is to be at x from the perspective of the player to move at the root of the tree.

b. A policy estimator $\hat{\pi} : \mathcal{T} \times \mathcal{X} \to \mathcal{P}(\mathcal{X})$, which returns a probability distribution over possible next states $\hat{\pi}_T(x)$ given a tree T and a node x in T. The policy estimator is meant to approximate the result of playing the optimal policy from x (from the perspective of the player to move at x).

For both KataGo and AlphaGo, the value function estimator and policy estimator are defined by two deep neural network heads with a shared backbone. The reason that $\hat{V}$ and $\hat{\pi}$ also take a tree T as an argument is because the estimators factor in the sequence of moves leading up to a node in the tree.

A playout is performed by taking a walk in the current game tree T. The walk goes down the tree until it attempts to walk to a node x' that either doesn't exist in the tree or is a terminal node.[†] At this point the playout ends and x' is added as a new node to the tree (we allow duplicate terminal nodes in the tree).

Walks start at the root of the tree. Let x be where we are currently in the walk. The child c we walk to (which may not exist in the tree) is given by

$$\begin{aligned}&\operatorname{walk}_T^{\mathrm{MCTS}}(x)\\&= \begin{cases} \underset{c}{\operatorname{argmax}} \quad \bar{V}_T(c) + \alpha \cdot \hat{\pi}_T(x)[c] \cdot \frac{\sqrt{S_T(x)-1}}{1+S_T(c)} & \text{if root player to move at } x, \\ \underset{c}{\operatorname{argmin}} \quad \bar{V}_T(c) - \alpha \cdot \hat{\pi}_T(x)[c] \cdot \frac{\sqrt{S_T(x)-1}}{1+S_T(c)} & \text{if opponent player to move at } x, \end{cases}\end{aligned} \tag{E.1}$$

where the argmin and argmax are taken over all children reachable in a single legal move from x. There are some new pieces of notation in Eq E.1. Here are what they mean:

1. $\bar{V}_T : \mathcal{X} \to \mathbb{R}$ takes a node x and returns the average value of $\hat{V}_T$ across all the nodes in the subtree of T rooted at x (which includes x). In the special case that x is a terminal node, $\bar{V}_T(x)$ is the result of the finished game as given by the game-simulator. When x does not exist in T, we instead use the more complicated formula[‡]

$$\bar{V}_T(x) = \bar{V}_T(\operatorname{par}_T(x)) - \beta \cdot \sqrt{\sum_{x' \in \operatorname{children}_T(\operatorname{par}_T(x))} \hat{\pi}_T(\operatorname{par}_T(x))[x']} \;,$$

where $\operatorname{par}_T(x)$ is the parent of x in T and β is a constant that controls how much we de-prioritize exploration after we have already done some exploration.

2. $\alpha \geqslant 0$ is a constant to trade off between exploration and exploitation.

3. $S_T : \mathcal{X} \to \mathbb{Z}_{\geqslant 0}$ takes a node x and returns the size of the subtree of T rooted at x. Duplicate terminal nodes are counted multiple times. If x is not in T, then $S_T(x) = 0$.

[†] A "terminal" node is one where the game is finished, whether by the turn limit being reached, one player resigning, or by two players passing consecutively.

[‡] Which is used in KataGo and LeelaZero but not AlphaGo [175].

The first term in Eq E.1 can be thought of as the exploitation term, and the second term can be thought of as the exploration term and is inspired by UCB algorithms.

E.2.1.2 MCTS Final Move Selection

The final move to be selected by MCTS is sampled from a distribution proportional to

$$S_{T_N}(c)^{1/\tau}, \tag{E.2}$$

where c in this case is a child of the root node. The temperature parameter τ trades off between exploration and exploitation.[§]

E.2.1.3 Efficiently Implementing MCTS

To efficiently implement the playout procedure one should keep running values of $\bar{V}_T$ and S_T for every node in the tree. These values should be updated whenever a new node is added. The standard formulation of MCTS bakes these updates into the algorithm specification. Our formulation hides the procedure for computing $\bar{V}_T$ and S_T to simplify exposition.

E.2.2 Adversarial MCTS: Sample (A-MCTS-S)

In this section, we describe in detail how our Adversarial MCTS: Sample (A-MCTS-S) attack is implemented. We build off of the framework for vanilla MCTS as described in Appendix E.2.1.

A-MCTS-S, just like MCTS, starts from a tree T_0 with a single root node and adds nodes to the tree via a series of N playouts. We derive the next move distribution from the final game tree T_N by sampling from the distribution proportional to

$$S^{\text{A-MCTS}}_{T_N}(c)^{1/\tau}, \quad \text{where } c \text{ is a child of the root node of } T_N. \tag{E.3}$$

Here, $S^{\text{A-MCTS}}_T$ is a modified version of S_T that measures the size of a subtree while ignoring non-terminal defender-nodes (at defender-nodes it is the defender's turn to move, and at self-nodes it is the adversary's turn to move). Formally, $S^{\text{A-MCTS}}_T(x)$ is the sum of the weights of nodes in the subtree of T rooted at x, with weight function

$$w^{\text{A-MCTS}}_T(x) = \begin{cases} 1 & \text{if } x \text{ is self-node,} \\ 1 & \text{if } x \text{ is terminal defender-node,} \\ 0 & \text{if } x \text{ is non-terminal defender-node.} \end{cases} \tag{E.4}$$

We grow the tree by A-MCTS playouts. At defender-nodes, we sample directly from the defender's policy π^v:

$$\text{walk}^{\text{A-MCTS}}_T(x) := \text{sample from } \pi^v_T(x). \tag{E.5}$$

[§] See `search.h::getChosenMoveLoc` and `searchresults.cpp::getChosenMoveLoc` to see how KataGo does this.

This is a perfect model of the defender *without* search. However, it will tend to underestimate the strength of the defender when the defender plays with search.

At self-nodes, we instead take the move with the best upper confidence bound just like in regular MCTS:

$$\text{walk}_T^{\text{A-MCTS}}(x) := \underset{c}{\text{argmax}} \quad \bar{V}_T^{\text{A-MCTS}}(c) + \alpha \cdot \hat{\pi}_T(x)[c] \cdot \frac{\sqrt{S_T^{\text{A-MCTS}}(x) - 1}}{1 + S_T^{\text{A-MCTS}}(c)}. \tag{E.6}$$

Note this is similar to Eq E.1 from the previous section. The key difference is that we use $S_T^{\text{A-MCTS}}(x)$ (a weighted version of $S_T(x)$) and $\bar{V}_T^{\text{A-MCTS}}(c)$ (a weighted version of $\bar{V}_T(c)$). Formally, $\bar{V}_T^{\text{A-MCTS}}(c)$ is the weighted average of the value function estimator $\hat{V}_T(x)$ across all nodes x in the subtree of T rooted at c, weighted by $w_T^{\text{A-MCTS}}(x)$. If c does not exist in T or is a terminal node, we fall back to the behavior of $\bar{V}_T(c)$.

E.2.3 Pass-Alive Defense

Our hard-coded defense modifies KataGo's C++ code to directly remove passing moves from consideration after MCTS, setting their probability to zero. Since the defender must eventually pass in order for the game to end, we allow passing to be assigned nonzero probability when there are no legal moves, *or* when the only legal moves are inside the defender's own pass-alive territory. We use a pre-existing function inside the KataGo codebase, `Board::calculateArea`, to determine which moves are in pass-alive territory.

The term "pass-alive territory" is defined in the KataGo rules as follows [177]:

> A {maximal-non-black, maximal-non-white} region R is *pass-alive-territory* for {Black, White} if all {black, white} regions bordering it are pass-alive-groups, and all or all but one point in R is adjacent to a {black, white} pass-alive-group, respectively.

The notion "pass-alive group" is a standard concept in Go [177]:

> A black or white region R is a *pass-alive-group* if there does not exist any sequence of consecutive pseudolegal moves of the opposing color that results in emptying R.

KataGo uses an algorithm introduced by Benson [20] to efficiently compute the pass-alive status of each group. For more implementation details, we encourage the reader to consult the official KataGo rules and the KataGo codebase on GitHub.

E.3 Hyperparameter Settings

We enumerate the key hyperparameters used in our training run in Table E.1. For brevity, we omit hyperparameters that are the same as KataGo defaults and have only a minor effect on performance.

Hyperparameter	**Value**	**Different from KataGo?**
Batch Size	256	Same
Learning Rate Scale of Hardcoded Schedule	1.0	Same
Minimum Rows Before Shuffling	250,000	Same
Data Reuse Factor	4	Similar
Adversary Visit Count	600	Similar
Adversary Network Architecture	`b6c96`	Different
Gatekeeping	Disabled	Different
Auto-komi	Disabled	Different
Komi randomization	Disabled	Different
Handicap Games	Disabled	Different
Game Forking	Disabled	Different

Table E.1: Key hyperparameter settings for our adversarial training runs.

The key difference from standard KataGo training is that our adversarial policy uses a `b6c96` network architecture, consisting of 6 blocks and 96 channels. This is much smaller than the defender, which uses a `b20c256` or `b40c256` architecture. We additionally disable a variety of game rule randomizations that help make KataGo a useful AI teacher in a variety of settings but are unimportant for our attack. We also disable gatekeeping, designed to stabilize training performance, as our training has proved sufficiently stable without it.

We train at most 4 times on each data row before blocking for fresh data. This is comparable to the original KataGo training run, although the ratio during that run varied as the number of asynchronous selfplay workers fluctuated over time. We use an adversary visit count of 600, which is comparable to KataGo, though the exact visit count has varied between their training runs.

E.3.1 Configuration for Curriculum Against Defender Without Search

In Section 9.5.1, we train using a curriculum over checkpoints, moving on to the next checkpoint when the adversary's win-rate exceeds 50%. We ran the curriculum over the following checkpoints, all without search:

1. Checkpoint 127: `b20c256x2-s5303129600-d1228401921` (`Initial`).

2. Checkpoint 200: `b40c256-s5867950848-d1413392747`.

3. Checkpoint 300: `b40c256-s7455877888-d1808582493`.

4. Checkpoint 400: `b40c256-s9738904320-d2372933741`.

5. Checkpoint 469: `b40c256-s11101799168-d2715431527`.

6. Checkpoint 505: `b40c256-s11840935168-d2898845681` (`Latest`).

These checkpoints can all be obtained from [176].

We start with checkpoint 127 for computational efficiency: it is the strongest KataGo network of its size, 20 blocks or `b20`. The subsequent checkpoints are all 40 block networks, and are approximately equally spaced in terms of training time steps. We include checkpoint 469 in between 400 and 505 for historical reasons: we ran some earlier experiments against checkpoint 469, so it is helpful to include checkpoint 469 in the curriculum to check performance is comparable to prior experiments.

Checkpoint 505 is the latest *confidently rated* network. There are some more recent, larger networks (`b60` = 60 blocks) that may have an improvement of up to 150 Elo. However, they have had too few rating games to be confidently evaluated.

E.4 Strength of Go AI systems

In this section, we estimate the strength of KataGo's `Latest` network with and without search nd the AlphaZero agent from [142] playing with 800 visits.

E.4.1 Strength of KataGo Without Search

First, we estimate the strength of KataGo's `Latest` agent playing without search. We use two independent methodologies and conclude that `Latest` without search is at the level of a weak professional.

One way to gauge the performance of `Latest` without search is to see how it fares against humans on online Go platforms. Per Table E.2, on the online Go platform KGS, a slightly earlier (and weaker) checkpoint than `Latest` playing without search is roughly at the level of a top-100 European player. However, some caution is needed in relying on KGS rankings:

1. Players on KGS compete under less focused conditions than in a tournament, so they may underperform.

2. KGS is a less serious setting than official tournaments, which makes cheating (e.g., using an AI) more likely. Thus human ratings may be inflated.

3. Humans can play bots multiple times and adjust their strategies, while bots remain static. In a sense, humans are able to run adversarial attacks on the bots, and are even able to do so in a white-box manner since the source-code and network weights of a bot like KataGo are public.

Another way to estimate the strength of `Latest` without search is to compare it to other AIs with known strengths and extrapolate performance across different amounts of search.

KGS handle	**Is KataGo?**	**KGS rank**	**EGF rank**	**EGD Profile**
Fredda		22	25	Fredrik Blomback
cheater		25	6	Pavol Lisy
TeacherD		26	39	Dominik Boviz
NeuralZ03	✓	31		
NeuralZ05	✓	32		
NeuralZ06	✓	35		
ben0		39	16	Benjamin Drean-Guenaizia
sai1732		40	78	Alexandr Muromcev
Tichu		49	64	Matias Pankoke
Lukan		53	10	Lukas Podpera
HappyLook		54	49	Igor Burnaevskij

Table E.2: Rankings of various humans and no-search KataGo bots on KGS [78]. Human players were selected to be those who have European Go Database (EGD) profiles [45], from which we obtained the European Go Federation (EGF) rankings in the table. The KataGo bots are running with a checkpoint slightly weaker than `Latest`, specifically Checkpoint 469 or `b40c256-s11101799168-d2715431527` [135]. Per [176], the checkpoint is roughly 10 Elo weaker than `Latest`.

Our analysis critically assumes the transitivity of Elo at high levels of play. We walk through our estimation procedure below:

1. Our anchor is ELF OpenGo at 80,000 visits per move, which won all 20 games played against four top-30 professional players, including five games against the now world number one [161]. We assume that ELF OpenGo at 80,000 visits is strongly superhuman, meaning it has a 90%+ winrate over the strongest current human.[¶] At the time of writing, the top ranked player on Earth has an Elo of 3845 on goratings.org [35]. Under our assumption, ELF OpenGo at 80,000 visits per move would have an Elo of 4245+ on goratings.org.

2. The strongest network in the original KataGo paper was shown to be slightly stronger than ELF OpenGo [175, Table 1] when both bots were run at 1600 visits per move. From Figure E.1, we see that the relative strengths of KataGo networks is maintained across different amounts of search. We thus extrapolate that KataGo at 80,000 visits would also have an Elo of 4245+ on goratings.org.

3. The strongest network in the original KataGo paper is comparable to the `b15c192-s1503689216-d402723070` checkpoint on katagotraining.org [176]. We dub this check-

[¶] This assumption is not entirely justified by statistics, as a 20:0 record only yields a 95% binomial lower confidence bound of a 83.16% win rate against top-30 professional players in 2019. It does help however that the players in question were rated #3, #5, #23, and #30 in the world at the time.

point `Original`. In a series of benchmark games, we found that `Latest` without search won 29/3200 games against `Original` with 1600 visits. This puts `Original` with 1600 visits ~800 Elo points ahead of `Latest` without search.

4. Finally, log-linearly extrapolating the performance of `Original` from 1600 to 80,000 visits using Figure E.1 yields an Elo difference of ~900 between the two visit counts.

5. Combining our work, we get that `Latest` without search is roughly 800 + 900 = ~1700 Elo points weaker than ELF OpenGo with 80,000 visits. This would give `Latest` without search an Elo rating of 4245 − 1700 = ~2500 on goratings.org, putting it at the skill level of a weak professional.

As a final sanity check on these calculations, the raw AlphaGo Zero neural network was reported to have an Elo rating of 3,055, comparable to AlphaGo Fan's 3,144 Elo.[‖] Since AlphaGo Fan beat Fan Hui, a 2-dan professional player [153], this confirms that well-trained neural networks can play at the level of human professionals. Although there has been no direct comparison between KataGo and AlphaGo Zero, we would expect them to be not wildly dissimilar. Indeed, if anything the latest versions of KataGo are likely stronger, benefiting from both a large distributed training run (amounting to over 10,000 V100 GPU days of training) and four years of algorithmic progress.

E.4.2 Strength of KataGo With Search

In the previous section, we established that `Latest` without search is at the level of a weak professional with rating around ~2500 on goratings.org.

Assuming Elo transitivity, we can estimate the strength of `Latest` by utilizing Figure E.1. In particular, our evaluation results tell us that `Latest` with 64 playouts/move is roughly 1063 Elo stronger than `Latest` with no search. This puts `Latest` with 64 playouts/move at an Elo of ~3563 on goratings.org—within the top 20 in the world.

E.4.3 Strength of AlphaZero

Prior work from Timbers et al. [162] described in Section 9.2 exploited the AlphaZero replica from Schmid et al. [142] playing with 800 visits. Unfortunately, this agent has never been evaluated against KataGo or against any human player, making it difficult to directly compare its strength to those of our defenders. Moreover, since it is a proprietary model, we cannot perform this evaluation ourselves. Accordingly, in this section we seek to estimate the strength of these AlphaZero agents using three anchors: GnuGo, Pachi and Lee Sedol. Our estimates suggest AlphaZero with 800 visits ranges in strength from the top 200 of human players, to being slightly superhuman.

[‖] The Elo scale used in Silver et al. [153] is not directly comparable to our Elo scale, although they should be broadly similar as both are anchored to human players.

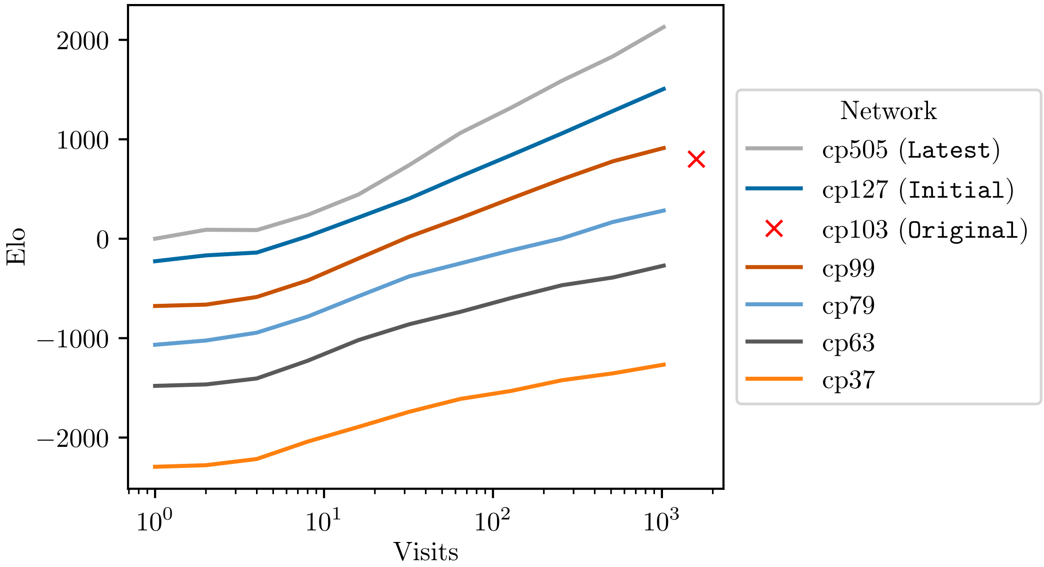


Figure E.1: Elo ranking (y-axis) of networks (different colored lines) by visit count (x-axis). The lines are approximately linear on a log x-scale, with the different networks producing similarly shaped lines vertically shifted. This indicates that there is a *consistent* increase in Elo, regardless of network strength, that is logarithmic in visit count. Elo ratings were computed from self-play games among the networks using a Bayesian Elo estimation algorithm [62].

Agent	Defender?	Elo (rel GnuGo)	Elo (rel defender)
AlphaZero(s=16k, t=800k)		+3139	+1040
AG0 3-day(s=16k)		+3069	+970
AlphaGo Lee(time=1s)		+2308	+209
AlphaZero(s=800,t=800k)	✓	**+2099**	0
Pachi(s=100k)		+869	-1230
Pachi(s=10k)		+231	-1868
GnuGo(l=10)		+0	-2099

Table E.3: Relative Elo ratings for AlphaZero, drawing on information from Schmid et al. [142, Table 4], Silver et al. [152] and Silver et al. [153]. s stands for number of steps, time for thinking time, and t for number of training steps.

We reproduce relevant Elo comparisons from prior work in Table E.3. In particular, Table 4 of Schmid et al. [142] compares the defender used in Timbers et al. [162], AlphaZero(s=800,t=800k), to two open-source AI systems, GnuGo and Pachi. It also compares it to a higher visit count version AlphaZero(s=16k, t=800k), from which we can compare using Silver et al. [152] to AG0 3-day and from there using Silver et al. [153] to AlphaGo Lee which played Lee Sedol.

Our first strength evaluation uses the open-source anchor point provided by Pachi(s=10k). The authors of Pachi [18] report it achieves a 2-dan ranking on KGS [17] when playing with 5000 playouts and using up to 15,000 when needed. We conservatively assume this corresponds to a 2-dan EGF player (KGS rankings tend to be slightly inflated), giving an EGF Elo of 2200. Assuming transitivity, the defender AlphaZero(s=800,t=800k) would then have an EGF Elo of 4299. The top EGF professional Ilya Shiskin has an EGF Elo of 2830 [46] at the time of writing, and 2979 on goratings.org [35]. Using Ilya as an anchor, this would give AlphaZero(s=800,t=800k) a goratings.org Elo of 4299+2979-2830=4448. This is superhuman, as the top player at the time of writing has an Elo of 3845.

However, some caution is needed here—the Elo gap between Pachi(s=10k) and AlphaZero(s=800,t=800k) is huge, making the exact value unreliable. The gap from Pachi(s=100k) is smaller, however unfortunately to the best of our knowledge there is no public evaluation of Pachi at this strength. However, the results in Baudiš and Gailly [17] strongly suggest it would perform at no more than a 4-dan KGS level, or at most an EGF Elo of 2400.[**] Repeating the analysis above then gives a goratings.org Elo of 2400+(2308-869)+(2979-2830)=3988 Elo. This still suggests the defender is superhuman, but only barely.

However, if we instead take GnuGo level 10 as our anchor, we get a quite different result. It is known to play between 10 and 11kyu KGS [77], or an EGF Elo of around 1050. This gives an implied EGF Elo for AlphaZero(s=800,t=800k) of 3149, or a goratings.org Elo of 3298 Elo. This is still strong, in the top 200 world players, but is far from superhuman.

The large discrepancy between these results led us to seek a third anchor point: how AlphaZero performed relative to previous AlphaGo models that played against humans. A complication is that the version of AlphaZero that Timbers et al. use differs from that originally reported in Silver et al. [152], however based on private communication with Timbers et al. we are confident the performance is comparable:

> These agents were trained identically to the original AlphaZero paper, and

[**] In particular, Baudiš and Gailly [17] report that Pachi achieves a 3-dan to 4-dan ranking on KGS when playing on a cluster of 64 machines with 22 threads, compared to 2-dan on a 6-core Intel i7. Figure 4 of Baudiš and Gailly [18] confirms playouts are proportional to the number of machines and number of threads, and we'd therefore expect the cluster to have 200x as many visits, or around a million visits. If 1 million visits is at best 4-dan, then 100,000 visits should be weaker. However, there is a confounder: the 1 million visits was distributed across 64 machines, and Figure 4 shows that distributed playouts do worse than playouts on a single machine. Nonetheless, we would not expect this difference to make up for a 10x difference in visits. Indeed, Baudiš and Gailly [18, Figure 4] shows that 1 million playouts spread across 4 machines (red circle) is substantially better than 125,000 visits on a single machine (black circle), achieving an Elo of around 150 compared to -20.

> were trained for the full 800k steps. We actually used the original code, and did a lot of validation work with Julian Schrittweiser & Thomas Hubert (two of the authors of the original AlphaZero paper, and authors of the ABR paper) to verify that the reproduction was exact. We ran internal strength comparisons that match the original training runs.

Table 1 of Silver et al. [152] shows that AlphaZero is slightly stronger than AG0 3-day (AlphaGo Zero, after 3 days of training), winning 60 out of 100 games giving an Elo difference of +70. This tournament evaluation was conducted with both agents having a thinking time of 1 minute. Table S4 from Silver et al. [152] reports that 16k visits are performed per second, so the tournament evaluation used a massive $960k$ visits–significantly more than reported on in Table E.3. However, from Figure E.1 we would expect the *relative* Elo to be comparable between the two systems at different visit counts, so we extrapolate AG0 3-day at 16k visits as being an Elo of $3139 - 70 = 3069$ relative to AlphaZero(s=16k, t=800k).

Figure 3a from Silver et al. [153] report that AG0 3-day achieves an Elo of around 4500. This compares to an Elo of 3,739 for AlphaGo Lee. To the best of our knowledge, the number of visits achieved per second of AlphaGo Lee has not been reported. However, we know that AG0 3-day and AlphaGo Lee were given the same amount of thinking time, so we can infer that AlphaGo Lee has an Elo of -761 relative to AG0 3-day. Consequently, AlphaGo Lee(time=1s) thinking for 1 second has an Elo relative to GnuGo of $3069 - 761 = 2308$.

Finally, we know that AlphaGo Lee beat Lee Sedol in four out of five matches, giving AlphaGo Lee a +240 Elo difference relative to Lee Sedol, or that Lee Sedol has an Elo of 2068 relative to Gnu Go level 10. This would imply that the defender is slightly stronger than Lee Sedol. However, this result should be taken with some caution. First, it relies on transitivity through many different versions of AlphaGo. Second, the match between AlphaGo Lee and Lee Sedol was played under two hours of thinking time with 3 byoyomi periods of 60 seconds per move Silver et al. [152, page 30]. We are extrapolating from this to some hypothetical match between AlphaGo Lee and Lee Sedol with only 1 second of thinking time per player. Although the Elo rating of Go AI systems seems to improve log-linearly with thinking time, it is unlikely this result holds for humans.

E.5 Experimental Results

E.5.1 Mimicking the Adversarial Policy

Our adversarial policies appear to follow a very simple strategy. They play in the corners and edges, staking out a small region of territory while allowing the defender to amass a larger territory. However, the adversary ensures that it is ahead in raw points prior to the defender securing its territory. If the defender then passes prematurely, the adversary wins.

However, it is possible that this seemingly simple policy hides a more nuanced exploit. For example, perhaps the pattern of stones it plays form an adversarial example for the

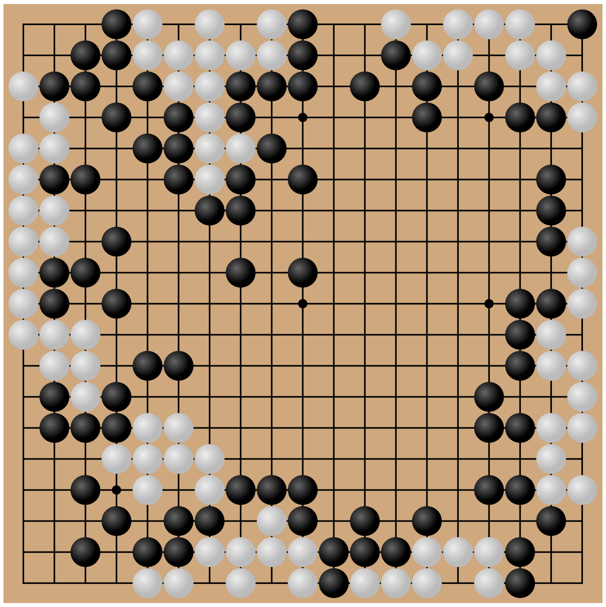


Figure E.2: An author of this paper plays as white mimicking our adversarial policy in a game against a KataGo-powered, 8-dan KGS rank bot `NeuralZ06` which has `friendlyPassOk` enabled. White wins by 18.5 points under Tromp-Taylor rules. See the full game.

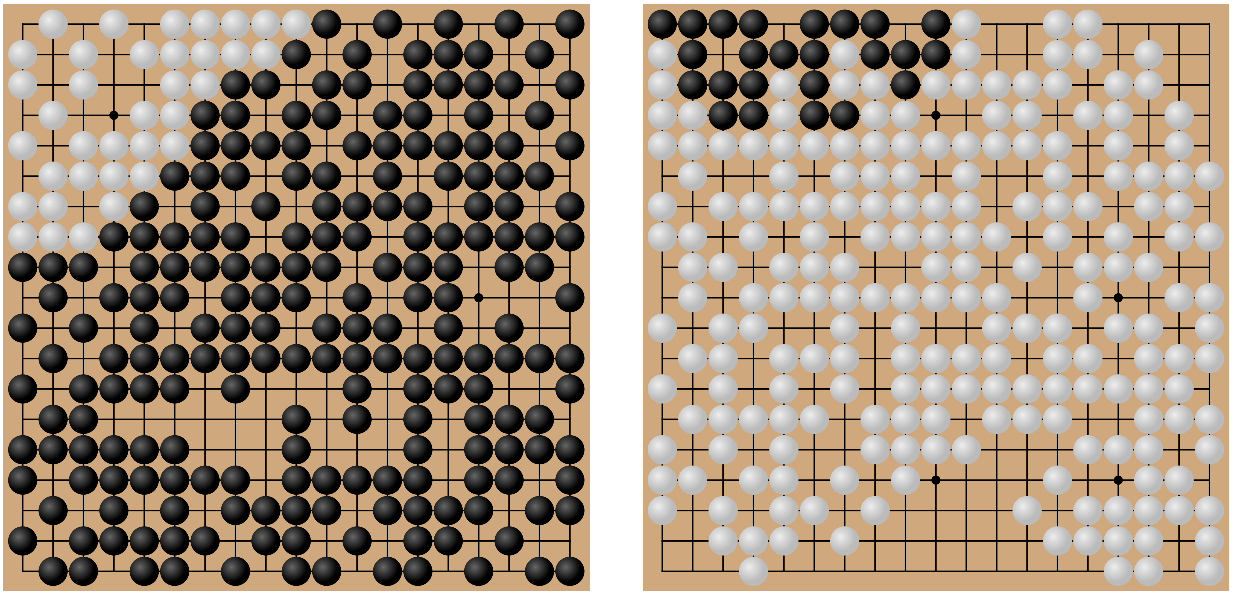
defender's network. To test this, one of the authors attempted to mimic the adversarial policy after observing some of its games.

The author was unable replicate this attack when KataGo was configured in the same manner as for the training and evaluation runs in this paper. However, when the `friendlyPassOk` flag in KataGo was turned on, the author was able successfully replicate this attack against the `NeuralZ06` bot on KGS, as illustrated in Figure E.2. This bot uses checkpoint 469 (see Appendix E.3.1) with no search. The author has limited experience in Go and is certainly weaker than 20 kyu, so they did not win due to any skill in Go.

E.5.2 Humans vs. Adversarial Policy

The same author from Section E.5.1 (strength weaker than 20kyu) played two manual games against the strongest adversary from Figure 9.3. In both games the author was able to achieve an overwhelming victory. See Figure E.3 for details.

This evaluation is imperfect in one significant way: the adversary was not playing with an accurate model of the author (rather it modeled the author as `Latest` with 1 visit). However, given our understanding of how the adversary works and the fact that the author in question knows not to prematurely pass, we predict that the adversary would probably not win even if it had access to an accurate model of the author.



(a) An author (B) defeats the strongest adversary (W). Explore the game.

(b) An author (W) defeats the strongest adversary (B). Explore the game.

Figure E.3: Two games between an author of this paper and the strongest adversary from Figure 9.3. In both games, the author achieves an overwhelming victory. The adversary used 600 playouts / move and used `Latest` as the model of its human opponent. The adversary used A-MCTS-S for one game and A-MCTS-S++ for the other.

E.5.3 Transferring to Other Checkpoints

In Figure E.4, we train adversaries against the `Latest` and `Initial` checkpoints respectively and evaluate against both checkpoints. We find adversaries do substantially better against the defender they were trained to target, although they do transfer to a limited extent.

E.5.4 Baseline Attacks

In Figure E.5, we plot the win margin of the KataGo defender `Latest` playing against baselines.

E.5.5 Adversarial Board State

This paper focuses on training an *agent* that can exploit Go-playing AI systems. A related problem is to find an adversarial *board state* which could be easily won by a human, but which Go-playing AI systems will lose from. In many ways this is a simpler problem, as an adversarial board state need not be a state that the defender agent would allow us to reach in normal play. Nonetheless, adversarial board states can be a useful tool to probe the blindspots that Go AI systems may have.

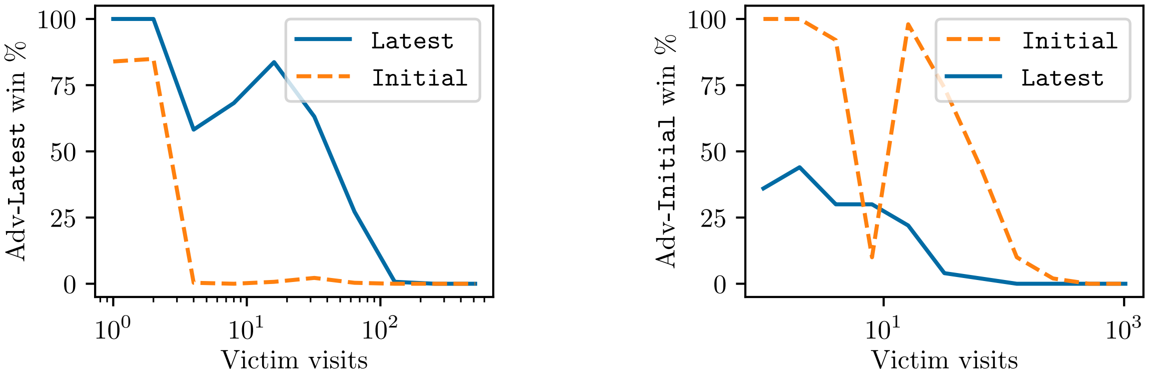


Figure E.4: An adversary trained against `Latest` (left) or `Initial` (right), evaluated against both `Latest` and `Initial` at various visit counts. The adversary always uses 600 visits / move.

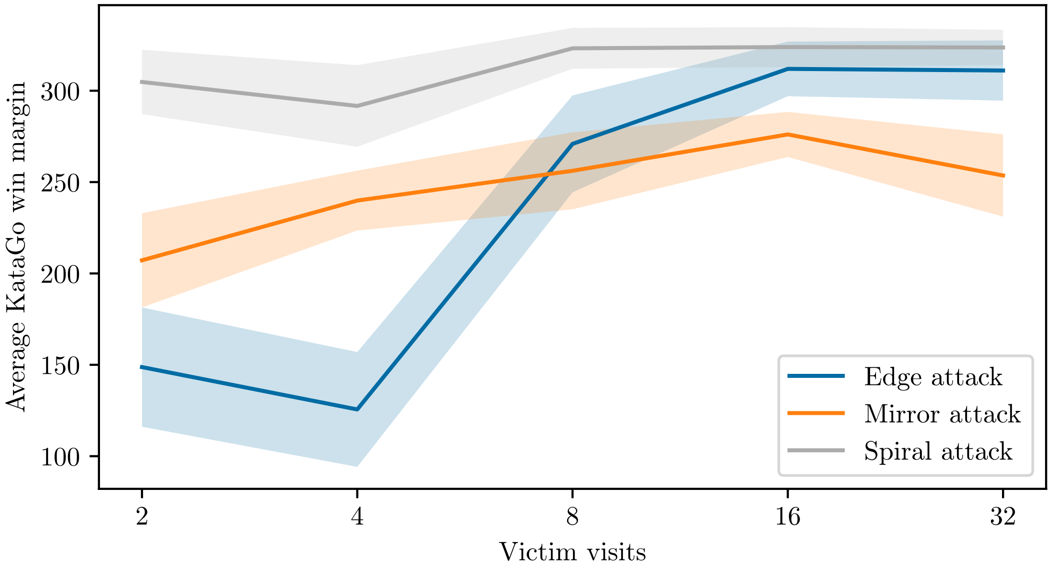


Figure E.5: The win margin of the `Latest` *defender* playing against baselines for varying defender visit counts (x-axis). Note the margin is positive indicating the defender on average gains more points than the baseline.

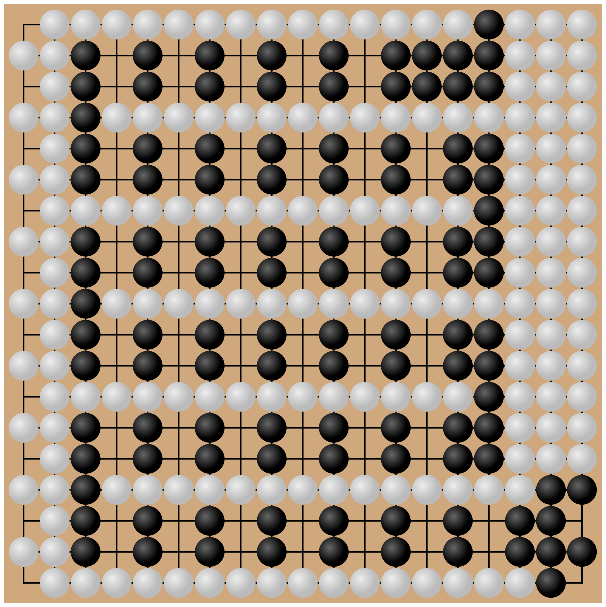


Figure E.6: A hand-crafted adversarial example for KataGo and other Go-playing AI systems. It is black's turn to move. Black can guarantee a win by connecting its currently disconnected columns together and then capturing the large white group on the right. However, KataGo playing against itself from this position loses 48% of the time—and 18% of the time it wins by a narrow margin of only 0.5 points!

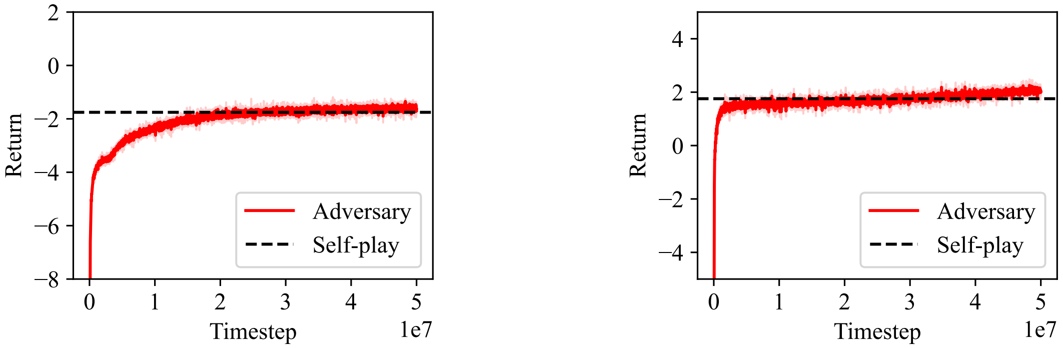
In Figure E.6 we present a manually constructed adversarial board state. Although quite unlike what would occur in a real game, it represents an interesting if trivial (for a human) problem. The black player can always win by executing a simple strategy. If white plays in between two of black's disconnected groups, then black should immediately respond by connecting those groups together. Otherwise, the black player can connect any two of its other disconnected groups together. Whatever the white player does, this strategy ensures that blacks' groups will eventually all be connected together. At this point, black has surrounded the large white group on the right and can capture it, gaining substantial territory and winning.

Although this problem is simple for human players to solve, it proves quite challenging for otherwise sophisticated Go AI systems such as KataGo. In fact, KataGo playing against a copy of itself *loses* as black 48% of the time. Even its wins are far less decisive than they should be—18% of the time it wins by only 0.5 points! We conjecture this is because black's winning strategy, although simple, must be executed flawlessly and over a long horizon. Black will lose if at any point it fails to respond to white's challenge, allowing white to fill in both empty spaces between black's groups. This problem is analogous to the classical cliff walking reinforcement learning task [157, Example 6.6].

Appendix F

Deferred content from Chapter 10

F.1 Adding a communication channel to Simple Push



(a) Agressor.

(b) Defender.

Figure F.1: Average return of agent controlling (a) aggressor and (b) defender in Simple Push *without* communication. The adversary, in red, fails to outperform the self-play baseline, in black.

Our initial experiments in the *Simple Push* environment did not lead to adversarial policies that were capable of outperforming their defenders (See Figures F.1a, F.1b).

Inspired by the cooperative environments explained in Lowe et al. [96], we add a communication channel: this channel allows each agent to observe a one-hot coded action taken by the other agent. This communication channel has no other effect on environment dynamics, and agents' reward does not depend on the contents of the communication channel. The size of the communication channel essentially represents the number of tokens either agent can use to communicate with the other. Because this setting is competitive, there is no reason for an agent to provide information in the communication channel which would be beneficial to the opponent. Therefore, an optimal policy should simply ignore the "messages" sent by the opponent. However, this channel increases the dimensionality and offers an adversary the possibility to learn messages that might "confuse" a (sub-optimal) defender.

In a small ablation on communication channels supporting 10, 25, 50 and 100 tokens, we find that adversarial policies are successful with 50 or more tokens and unsuccessful at less than 25. We also find that the number of timesteps until convergence, as well as general instability during training increases with higher sizes. For further experiments we use a communication channel of 50 tokens to allow for fast training while still providing an environment in which adversarial policies are possible.