Learning Safety Constraints from Demonstrations with Unknown Rewards

David Lindner
ETH Zurich

Xin Chen
ETH Zurich

Sebastian Tschiatschek
University of Vienna

Katja Hofmann
Microsoft Research, Cambridge

Andreas Krause
ETH Zurich

Abstract

We propose *Convex Constraint Learning for Reinforcement Learning* (`CoCoRL`), a novel approach for inferring shared constraints in a Constrained Markov Decision Process (CMDP) from a set of safe demonstrations with possibly different reward functions. While previous work is limited to demonstrations with known rewards or fully known environment dynamics, `CoCoRL` can learn constraints from demonstrations with *different unknown rewards* without knowledge of the environment dynamics. `CoCoRL` constructs a convex safe set based on demonstrations, which provably guarantees safety even for potentially sub-optimal (but safe) demonstrations. For near-optimal demonstrations, `CoCoRL` converges to the true safe set with no policy regret. We evaluate `CoCoRL` in gridworld environments and a driving simulation with multiple constraints. `CoCoRL` learns constraints that lead to safe driving behavior. Importantly, we can safely transfer the learned constraints to different tasks and environments. In contrast, alternative methods based on Inverse Reinforcement Learning (IRL) often exhibit poor performance and learn unsafe policies.

1 Introduction

Constrained Markov Decision Processes (CMDPs) integrate safety constraints with reward maximization in reinforcement learning (RL). However, similar to reward functions, it can be difficult to specify constraint functions manually (Krakovna et al., 2020). To tackle this issue, recent work proposes learning models of the constraints (e.g., Scobee and Sastry, 2020; Malik et al., 2021) analogously to reward models (Leike et al., 2018). However, existing approaches to constraint inference rely on demonstrations with *known reward functions*, which is often unrealistic and limiting in real-world applications.

For instance, consider the domain of autonomous driving (Figure 1), where specifying rewards/constraints is particularly difficult (Knox et al., 2023). However, we can gather diverse human driving trajectories that satisfy shared constraints, such as keeping an appropriate distance from other cars and avoiding crashes. These trajectories will usually have different (unknown) routes or driving style preferences, which we can model as different reward functions. In such scenarios, we aim to infer constraints from demonstrations with *shared constraints* but *unknown rewards*.

Contributions. In this paper: (1) we introduce the problem of inferring constraints in CMDPs from demonstrations with unknown rewards (Section 3.1); (2) we introduce *Convex Constraint Learning for Reinforcement Learning* (`CoCoRL`), a novel method for addressing this problem (Section 4); (3) we prove that `CoCoRL` guarantees safety (Section 4.1), and, for (approximately) optimal demonstrations, asymptotic optimality (Section 4.2), while IRL provably cannot guarantee safety (Section 3.2); and, (4) we conduct comprehensive empirical evaluations of `CoCoRL` in tabular environments and a continuous driving task with multiple constraints (Section 6). `CoCoRL` learns constraints that lead to safe driving behavior and that can be transferred across different tasks and environments. In contrast, methods based on Inverse Reinforcement Learning (IRL) often perform poorly and lack safety guarantees, and methods based on Imitation Learning (IL) lack robustness and transferability.

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

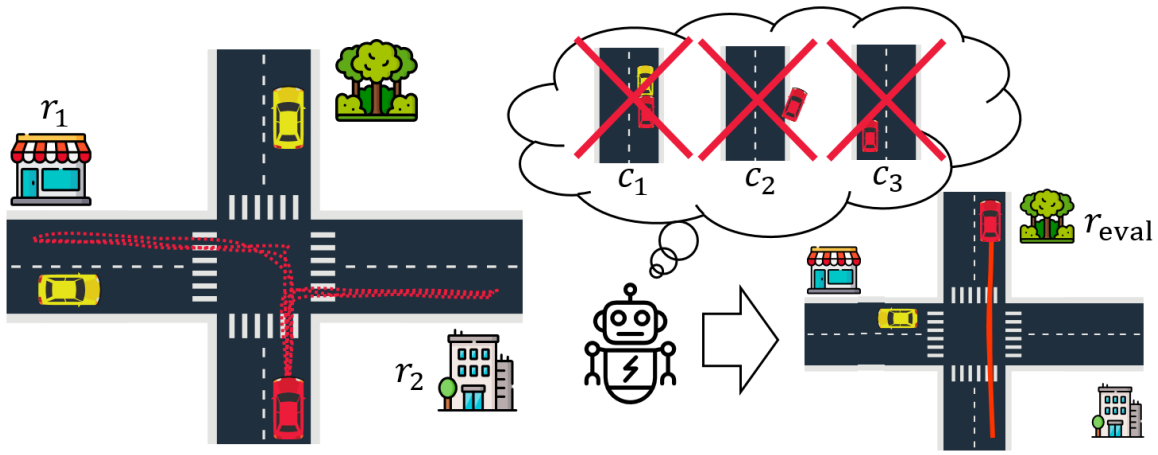


Figure 1: CoCoRL can, e.g., learn safe driving behavior from diverse driving trajectories with different unknown reward functions (here, r_1: "turn left", r_2: "turn right") . It infers constraints c_1, c_2, c_3 describing desirable driving behavior from demonstrations without knowledge of the specific reward functions r_1, r_2. These inferred constraints allow to optimize for a new reward function r_{eval} ("go straight"), ensuring safe driving behavior even in *new situations* where matching demonstrations are unavailable.

Overall, our theoretical and empirical results illustrate the potential of CoCoRL for constraint inference and the advantage of employing CMDPs with inferred constraints instead of relying on MDPs with IRL in safety-critical applications. We provide an open-source implementation of CoCoRL and the code necessary to reproduce all experiments at https://github.com/lasgroup/cocorl.

2 Background

Markov Decision Processes (MDPs). An (infinite-horizon, discounted) MDP (Puterman, 1990) is a tuple $(S, A, P, \mu_0, \gamma, r)$, where S is the state space, A the action space, $P : S \times A \times S \to [0, 1]$ the transition model, $\mu_0 : S \to [0, 1]$ the initial state distribution, $\gamma \in (0, 1)$ the discount factor, and $r : S \times A \to [0, 1]$ the reward function. An agent starts in a state $s \sim \mu_0$, and takes actions sampled from a policy $a \sim \pi(a|s)$ that lead to a next state according to the transition kernel $P(s'|s, a)$. The agent aims to maximize the expected discounted return $G_r(\pi) = \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. We often use the (discounted) occupancy measure $\mu_\pi(s, a) = \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{\{s_t=s, a_t=a\}}\right]$. For finite state and action spaces, the return is linear in μ_π, i.e., $G_r(\pi) = \sum_{s,a} \mu_\pi(s, a) r(s, a)$.

Constrained MDPs (CMDPs). A CMDP (Altman, 1999) extends an MDP with a set of cost functions $c_1, \ldots, c_n$ defined similar to the reward function as $c_j : S \times A \to [0, 1]$ and thresholds $\xi_1, \ldots, \xi_n$. The agent's goal is to maximize $G_r(\pi)$ under constraints on the expected discounted cumulative costs $J_j(\pi) = \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t c_j(s_t, a_t)\right]$. In particular, a feasible policy must satisfy $J_j(\pi) \leq \xi_j$ for all j. Again, we can write the cumulative costs as $J_j(\pi) = \sum_{s,a} \mu_\pi(s, a) c_j(s, a)$.

Inverse Reinforcement Learning (IRL). IRL aims to recover an MDP's reward function from expert demonstrations, ensuring the demonstrations are optimal under the identified reward function. It seeks a reward function where the optimal policy aligns with the expert demonstrations' feature expectations (Abbeel and Ng, 2004). Since the IRL problem is underspecified, algorithms differ in selecting a unique solution, with popular approaches including maximum margin IRL (Ng et al., 2000) and maximum (causal) entropy IRL (Ziebart et al., 2008).

3 Problem Setup and IRL-based Approaches

In this section, we first introduce the problem of inferring constraints from demonstrations in a CMDP. Then, we discuss why naive solutions based on IRL fail to solve this problem.

3.1 Problem Setup

We consider an infinite-horizon discounted CMDP without reward function and constraints, denoted by (S, A, P, μ_0, γ). We have access to demonstrations from k safe, i.e., feasible, policies $\mathcal{D} = \{\pi_1^*, \ldots, \pi_k^*\}$. The demonstrations have k unknown reward func-

tions $r_1, \dots, r_k$, but share the same constraints defined by unknown cost functions $c_1, \dots, c_n$ and thresholds $\xi_1, \dots, \xi_n$. We assume policy π_i^* is feasible in the CMDP $(S, A, P, \mu_0, \gamma, r_i, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$. Generally, we assume policy π_i^* optimizes for reward r_i, but it does not need to be optimal.

In our driving example (Figure 1), the rewards of the demonstrations can correspond to different routes and driver preferences, whereas the shared constraints describe driving rules and safety-critical behaviors, such as maintaining an appropriate distance to other cars or staying in the correct lane.

We assume that cost functions are linear in a d-dimensional feature space,[1] represented by $\mathbf{f} : S \times A \to [0,1]^d$. We represent demonstrations by their discounted feature expectations (slightly overloading notation) defined as $\mathbf{f}(\pi) = \mathbb{E}_{P,\pi}[\sum_{t=0}^{\infty} \gamma^t \mathbf{f}(s_t, a_t)]$. We then assume the cost functions are $c_j(s,a) = \phi_j^T \mathbf{f}(s,a)$, where each cost function has a parameter vector $\phi_j \in \mathbb{R}^d$. For now, we assume we know the exact feature expectations of the demonstrations. In Section 4.3, we discuss sample-based estimation. Our approach does not require linear reward functions in general.

For evaluation, we receive a new reward function r_{eval} and aim to find an optimal policy for the CMDP $(S, A, P, \mu_0, \gamma, r_{\text{eval}}, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$ where the constraints are unknown. Our goal is to ensure safety while achieving high reward under r_{eval}.

3.2 Limitations of IRL in CMDPs

One might be tempted to disregard the CMDP nature of the problem and try to apply standard IRL methods to infer a reward from demonstrations, and much prior work on constraint learning primarily extends work from IRL (e.g., Scobee and Sastry (2020); Stocking et al. (2021); Malik et al. (2021)). However, IRL poses at least two key problems in our setting: (1) IRL alone cannot guarantee safety, and (2) it is unclear how to learn a transferable representation of the constraints.

To elaborate, let us first assume we have a single expert demonstration from a CMDP and apply IRL to infer a reward function, assuming no constraints. Then, in some CMDPs, any reward functions IRL can infer can yield unsafe optimal policies.

Proposition 1 (IRL can be unsafe)**.** *There are CMDPs $\mathcal{C} = (S, A, P, \mu_0, \gamma, r, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$ such that for any optimal policy π^* in $\mathcal{C}$ and any reward function r_{IRL} that could be returned by an IRL algorithm, the resulting MDP $(S, A, P, \mu_0, \gamma, r_{IRL})$ has optimal policies that are unsafe in $\mathcal{C}$.*

This follows from CMDPs having only stochastic optimal policies, and MDPs always having a deterministic optimal policy (Puterman, 1990; Altman, 1999).[2]

Next let us consider our actually problem setting, where we have a set of demonstrations that share constraints. If we use IRL, it is unclear how to learn the shared part and transfer it to a new task. A first approach could be to learn as a *shared* reward penalty. Unfortunately, it turns out learning a shared penalty can be infeasible, even with known rewards.

Proposition 2. *Let $\mathcal{C} = (S, A, P, \mu_0, \gamma, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$ be a CMDP without reward. Let r_1, r_2 be two reward functions and π_1^* and π_2^* corresponding optimal policies in $\mathcal{C} \cup \{r_1\}$ and $\mathcal{C} \cup \{r_2\}$. Let $\mathcal{M} = (S, A, P, \mu_0, \gamma)$ be the corresponding MDP without reward. Without additional assumptions, we cannot guarantee the existence of a function $\hat{c} : S \times A \to \mathbb{R}$ such that π_1^* is optimal in the MDP $\mathcal{M} \cup \{r_1 + \hat{c}\}$ and π_2^* is optimal in the MDP $\mathcal{M} \cup \{r_2 + \hat{c}\}$.*

Similar to Proposition 1, this result stems from the fundamental difference between MDPs and CMDPs. These findings suggest that IRL is not a suitable approach for learning from demonstrations in a CMDP. Our empirical findings in Section 6 confirm this.

4 Convex Constraint Learning for Reinforcement Learning (CoCoRL)

We are now ready to introduce the CoCoRL algorithm. In essence, CoCoRL consists of three steps:

1. Construct a conservative safe set $\mathcal{S}$ from the demonstrations $\mathcal{D}$.
2. Construct an *inferred CMDP* from the original CMDP and the safe set $\mathcal{S}$.
3. Use a standard constrained RL algorithm to solve the inferred CMDP and return resulting policy.

In this section, we (1) introduce the key idea of using convexity to construct provably conservative safe set and the inferred CMDP (Section 4.1); (2) establish convergence results under (approximately) optimal demonstrations (Section 4.2); (3) introduce several approaches for using estimated feature expectations (Section 4.3); and, (4) discuss practical considerations when implementing CoCoRL (Section 4.4).

Importantly, to construct the conservative safe set, we only require safe demonstrations. For convergence, we make certain optimality assumptions on the

[1] Note that in the case of a finite state and action space, this assumption is *w.l.o.g.*, because we can use the state-action occupancy measure $\mu_\pi(s,a)$ as features.

[2] See Appendix A for a full proof of this and other theoretical results in this paper.

demonstrations, either exact optimality or Boltzmann-rationality. Appendix A contains all omitted proofs.

4.1 Constructing a Convex Safe Set

Let the *true safe set* $\mathcal{F} = \{\pi | \forall j : J_j(\pi) \leq \xi_j\}$ be the set of all policies that satisfy the constraints. `CoCoRL` is based on the key observation that $\mathcal{F}$ is convex.

Lemma 1 ($\mathcal{F}$ is convex)**.** *For any CMDP, suppose* $\pi_1, \pi_2 \in \mathcal{F} = \{\pi | \forall j : J_j(\pi) \leq \xi_j\}$. *Let* $\bar{\pi}_{12}$ *be a mixture policy such that* $\mathbf{f}(\bar{\pi}_{12}) = \lambda \mathbf{f}(\pi_1) + (1-\lambda)\mathbf{f}(\pi_2)$ *with* $\lambda \in [0, 1]$. *Then, we have* $\bar{\pi}_{12} \in \mathcal{F}$.

We know that all demonstrations are safe in the original CMDP and convex combinations of their feature expectations are safe. This insight leads us to a natural approach for constructing a conservative safe set: create the convex hull of the feature expectations of the demonstrations:

$$\mathcal{S} := \left\{ \pi \middle| \mathbf{f}(\pi) = \sum_{i=1}^{k} \lambda_i \mathbf{f}(\pi_i^*),\ \lambda_i \geq 0 \text{ and } \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

Thanks to the convexity of $\mathcal{F}$, we can now guarantee safety for all policies $\pi \in \mathcal{S}$.

Theorem 1 (Estimated safe set)**.** *Any policy* $\pi \in \mathcal{S}$ *is safe, i.e.,* $\pi \in \mathcal{F}$.

During evaluation, we want to find an optimal policy in our coservative safe set for a new reward function r_{eval}, i.e., we want to solve $\max_{\pi \in \mathcal{S}} G_{r_{\text{eval}}}(\pi)$. Conveniently, we can reduce this problem to solving a new CMDP with identical dynamics. Specifically, we can find a set of linear constraints to represent the safe set $\mathcal{S}$. This result follows from standard convex analysis: $\mathcal{S}$ is a convex polyhedron, which is the solution to a set of linear equations. Constructing a convex hull from a set of points and identifying the linear equations is a standard problem in polyhedral combinatorics.

Consequently, we can optimize within our safe set $\mathcal{S}$ by solving an "inferred" CMDP.

Theorem 2 (Inferred CMDP)**.** *We can find cost functions* $\{\hat{c}_j\}_{j=1}^m$ *and thresholds* $\{\hat{\xi}_j\}_{j=1}^m$ *such that for any reward function* r_{eval}, *solving the inferred CMDP* $(S, A, P, \mu_0, \gamma, r_{eval}, \{\hat{c}_j\}_{j=1}^m, \{\hat{\xi}_j\}_{j=1}^m)$ *is equivalent to finding* $\pi^* \in \operatorname{argmax}_{\pi \in \mathcal{S}} G_{r_{eval}}(\pi)$. *Consequently, if* $\mathcal{S}$ *contains an optimal policy for the true CMDP, solving the inferred CMDP returns a policy that is also optimal in the true CMDP.*

Note that the number of inferred constraints m is not necessarily equal to the number of true constraints n, but we do not assume to know n. It immediately follows from Theorem 2 that our safe set is worst-case optimal because, without additional assumptions, we cannot dismiss the possibility that the true CMDP coincides with our inferred CMDP.

Corollary 1 ($\mathcal{S}$ is maximal)**.** *If* $\pi \notin \mathcal{S}$, *there exist* $r_1, \ldots, r_k$ *and* $c_1, \ldots, c_n$, *such that the expert policies* $\pi_1^*, \ldots, \pi_k^*$ *are optimal in the CMDPs* $(S, A, P, \mu_0, \gamma, r_i, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$, *but* $\pi \notin \mathcal{F}$.

In essence, $\mathcal{S}$ is the largest set we can choose while ensuring safety.

4.2 Convergence for (Approximately) Optimal Demonstrations

So far, we studied the safety of `CoCoRL`. In this section, we aim to establish its convergence. To this end, we compare the solution returned by `CoCoRL`, i.e., $\max_{\pi \in \mathcal{S}} G_r(\pi)$, to the optimal safe solution, i.e., $\max_{\pi \in \mathcal{F}} G_r(\pi)$, for a given evaluation reward r_{eval}. In particular, we define the *policy regret*

$$\mathcal{R}(r, \mathcal{S}) = \max_{\pi \in \mathcal{F}} G_r(\pi) - \max_{\pi \in \mathcal{S}} G_r(\pi).$$

Let us assume the reward functions of the demonstrations and the reward functions during evaluation follow the same distribution $P(r)$. After observing k demonstrations, we construct the safe set $\mathcal{S}_k$, and consider the expectation $\mathbb{E}_r[\mathcal{R}(r, \mathcal{S}_k)]$ under the reward distribution. Now, we aim to show that regret is zero asymptotically, i.e., $\lim_{k \to \infty} \mathbb{E}_r\left[\mathcal{R}(r, \mathcal{S}_k)\right] = 0$.

In order to establish convergence for `CoCoRL`, we need additional assumptions. In particular, we need the demonstrations to be diverse enough to ensure we learn a sufficiently large safe set. One way to get this diversity is by assuming the demonstrations are (approximately) optimal for a set of *unknown* reward function.

In particular, we establish `CoCoRL`'s convergence to optimality under two conditions: (1) when the safe demonstrations are exactly optimal, and (2) when they are approximately optimal according to a Boltzmann model. To simplify the convergence analysis, we additionally assume the reward function are linear.[3] Importantly, we need these assumptions only for the convergence analysis in this section. They are not necessary for `CoCoRL` to guarantee safety (see Section 4.1) or work in practice (see Section 6).

First, let us consider exactly optimal demonstrations.

Assumption 1 (Linear rewards)**.** The demonstrations reward functions and evaluation reward functions are linear. In particular $r_i(s, a) = \theta_i^T \mathbf{f}(s, a)$ and $r_{\text{eval}}(s, a) = \theta_{\text{eval}}^T \mathbf{f}(s, a)$, where $\theta_i \in \mathbb{R}^d$ and $\theta_{\text{eval}}^T \in \mathbb{R}^d$.

[3] The linear reward assumption could likely be relaxed with more careful analysis, and the analysis could be extended to other types of optimality assumptions.

Assumption 2 (Exact optimality)**.** Each π_i^* is optimal in the CMDP $(S, A, P, \mu_0, \gamma, r_i, \{c_j\}_j, \{\xi_j\}_j)$.

Theorem 3 (Convergence, exact optimality)**.** *Under Assumptions 1 and 2, for any $\delta > 0$, after $k \geq \log(\delta/f_v(d,n))/\log(1-\delta/f_v(d,n))$, we have $P(\mathcal{R}(r,\mathcal{S}_k) > 0) \leq \delta$, where $f_v(d,n)$ is an upper bound on the number of vertices of the true safe set. In particular, $\lim_{k\to\infty} \mathbb{E}_r[\mathcal{R}(r,\mathcal{S}_k)] = 0$.*

Due to both the true safe set and the estimated safe set being convex polyhedra, we have a finite hypothesis class consisting of all convex polyhedra with vertices that are a subset of the true safe set's vertices. The proof builds on the insight that all vertices supported by the distribution induced by $P(r)$ will eventually be observed, while unsupported vertices do not contribute to the regret. The number of vertices is typically on the order $f_v(d,n) \in \mathcal{O}(n^{\lfloor d/2 \rfloor})$. A tighter bound is given, for example, by McMullen (1970).

In practical settings, it is unrealistic to assume perfectly optimal demonstrations. Thus, we relax this assumption and consider the case where demonstrations are only approximately optimal.

Assumption 3 (Boltzmann-rationality)**.** We observe demonstrations following a Boltzmann policy distribution $\pi_i^* \sim \exp(\beta G_{r_i}(\pi)) \mathbb{1}_{\{J_1(\pi)\leq\xi_1,\dots,J_n(\pi)\leq\xi_n\}}/Z_i$, where $G_{r_i}(\pi)$ is the expected return computed w.r.t. the reward r_i, and Z_i is a normalization constant.

Theorem 4 (Convergence, Boltzmann-rationality)**.** *Under Assumption 3, for any $\delta > 0$, after $k \geq \log(\delta/f_v(d,n))/\log(1-\exp(-\beta d/(1-\gamma)))$, we have $P(\mathcal{R}(r,\mathcal{S}_k) > 0) \leq \delta$, where $f_v(d,n)$ is an upper bound on the number of vertices of the true safe set. In particular, $\lim_{k\to\infty} \mathbb{E}_r[\mathcal{R}(r,\mathcal{S}_k)] = 0$.*

This result relies on the Boltzmann distribution having a fixed lower bound over the bounded set of policy features. By enumerating the vertices of the true safe set, we can establish an upper bound on the probability of the "bad event" where a vertex is not adequately covered by the set of k demonstrations. This shows how the regret must decrease as k increases.

4.3 Estimating Feature Expectations

So far, we assumed access to the true feature expectations of the demonstrations $\mathbf{f}(\pi_i^*)$. However, in practice, we often rely on samples from the environment to estimate the feature expectations. Given n_{traj} samples from policy π_i^*, we can estimate the feature expectations by taking the sample mean: $\hat{\mathbf{f}}(\pi_i^*) = \frac{1}{n_{\text{traj}}} \sum_{p=1}^{n_{\text{traj}}} \mathbf{f}(\tau_p^i)$, where $\mathbf{f}(\tau_p^i) = \sum_{t=0}^{\infty} \gamma^t \mathbf{f}(s_t, a_t)$ and $\tau_p^i = (s_0, a_0, s_1, a_1, \dots)$ represents a trajectory from rolling out π_i^* in the environment. This estimate is unbiased, i.e., $\mathbb{E}[\hat{\mathbf{f}}(\pi_i^*)] = \mathbf{f}(\pi_i^*)$, and we can use standard concentration inequalities to quantify its uncertainty. This allows us to ensure ϵ-safety when using the estimated feature expectations.

Theorem 5 (ϵ-safety with estimated feature expectations.)**.** *Suppose we estimate the feature expectations of each policy π_i^* using at least $n_{traj} > d\log(nk/\delta)/(2\epsilon^2(1-\gamma))$ samples, and construct the estimated safe set $\hat{\mathcal{S}} = \texttt{conv}(\hat{\mathbf{f}}(\pi_1^*), \dots, \hat{\mathbf{f}}(\pi_k^*))$. Then, we have $P(\max_j(J_j(\pi) - \xi_j) > \epsilon | \pi \in \hat{\mathcal{S}}) < \delta$.*

We can extend our analysis to derive convergence bounds for exactly optimal or Boltzmann-rational demonstrations, similar to Theorems 3 and 4. The exact bounds are provided in Appendix A.4. Importantly, we get no regret asymptotically in both cases.

Alternatively, we could leverage confidence sets around $\hat{\mathbf{f}}(\pi_i^*)$ to construct a conservative safe set. This approach guarantees exact safety and convergence as long as the confidence intervals shrink. However, it involves constructing a *guaranteed hull* (Sember, 2011), which can be computationally more expensive than constructing a simple convex hull. The guaranteed hull can also be overly conservative in practice. See Appendix B for a detailed discussion.

Crucially, `CoCoRL` can be applied in environments with continuous state and action spaces and nonlinear policy classes, as long as the feature expectations can be computed or estimated.

4.4 Practical Implementation

To implement `CoCoRL`, we first compute or estimate the demonstrations' feature expectations $\mathbf{f}(\pi_i^*)$. Then, we use the Quickhull algorithm (Barber et al., 1996) to construct a convex hull. Finally, we solve the inferred CMDP from Theorem 2 using a constrained RL algorithm, the choice depending on the environment. Constructing the convex hull only has negligible computational cost compared to solving the inferred CMDP. For very high-dimensional environments or large numbers of demonstrations it may become beneficial to approximate the convex hull instead.

We make two further extensions to enhance the robustness of this basic algorithm: we use projections to handle degenerate safe sets and we incrementally expand the safe set. We discuss them briefly here, and in Appendix D in more detail.

Handling degenerate safe sets. The demonstrations might lie in a lower-dimensional subspace of $\mathbb{R}^d$ (i.e., have a rank less than d). In that case, we project them onto this lower-dimensional subspace, construct the convex hull in that space, and then project back

the resulting convex hull to $\mathbb{R}^d$. This is beneficial because Quickhull is not specifically designed to handle degenerate convex hulls.

Incrementally expand the safe set. If we construct $\mathcal{S}$ from all demonstrations, it will have many redundant vertices, which can result in numerical instabilities. To avoid this, we incrementally add points from $\mathcal{D}$ to $\mathcal{S}$. We greedily add the point furthest away from $\mathcal{S}$ until the distance of the furthest point is less than some $\varepsilon > 0$. In addition to mitigating numerical issues, this approach reduces the number of constraints in the inferred CMDP, making it easier to solve. Importantly, we do not loose any safety guarantees.

5 Related Work

Constraint learning in RL. Previous work on safe RL typically assumes fully known safety constraints (e.g., see the review by Garcıa and Fernández, 2015). More recent research on *constraint learning* addresses this limitation. Scobee and Sastry (2020) and Stocking et al. (2021) use maximum likelihood estimation to learn constraints from demonstrations with known rewards in tabular and continuous environments, respectively. Malik et al. (2021) propose a more scalable algorithm based on maximum entropy IRL, and extensions to using maximum causal entropy IRL have been explored subsequently (Glazier et al., 2021; McPherson et al., 2021; Baert et al., 2023). Papadimitriou et al. (2022) adapt Bayesian IRL to learning constraints via a Lagrangian formulation. However, all of these approaches assume full knowledge of the demonstrations' reward functions. In contrast, Chou et al. (2020) propose a method that allows for parametric uncertainty about the rewards, but it requires fully known environment dynamics and does not apply to general CMDPs.

Multi-task IRL. Multi-task IRL addresses the problem of learning from demonstrations in different but related tasks. Some methods reduce multi-task IRL to multiple single-task IRL problems (Babes et al., 2011; Choi and Kim, 2012), but they do not fully leverage the similarity between demonstrations. Others treat multi-task IRL as a meta-learning problem, focusing on quickly adapting to new tasks (Dimitrakakis and Rothkopf, 2012; Xu et al., 2019; Yu et al., 2019; Wang et al., 2021). Amin et al. (2017) study *repeated IRL* where the reward for different tasks is split into a task-specific component and a shared component, similar to the shared constraints in our setting. However, in general, IRL-based methods are challenging to adapt to CMDPs where safety is crucial (see Section 3.2).

Learning constraints for driving behavior. In the domain of autonomous driving, inferring constraints from demonstrations has attracted significant attention due to safety concerns. Rezaee and Yadmellat (2022) extend the method by Scobee and Sastry (2020) with a VAE-based gradient descent optimization to learn driving constraints. Liu et al. (2023) propose a benchmark for constraint learning based on highway driving trajectories, and Gaurav et al. (2023) evaluate their constraint learning method on a similar dataset. However, all these approaches assume fully known reward functions, which is often unrealistic beyond basic highway driving scenarios.

Bandits & learning theory. Our approach draws inspiration from constraint inference in other fields. For instance, learning constraints has been studied in best-arm identification in linear bandits with known rewards (Lindner et al., 2022; Camilleri et al., 2022). Our problem is also related to one-class classification (e.g., Khan and Madden, 2014), the study of random polytopes (e.g., Baddeley et al., 2007), and PAC-learning of convex polytopes (e.g., Gottlieb et al., 2018). However, none of these methods directly apply to the structure of our problem.

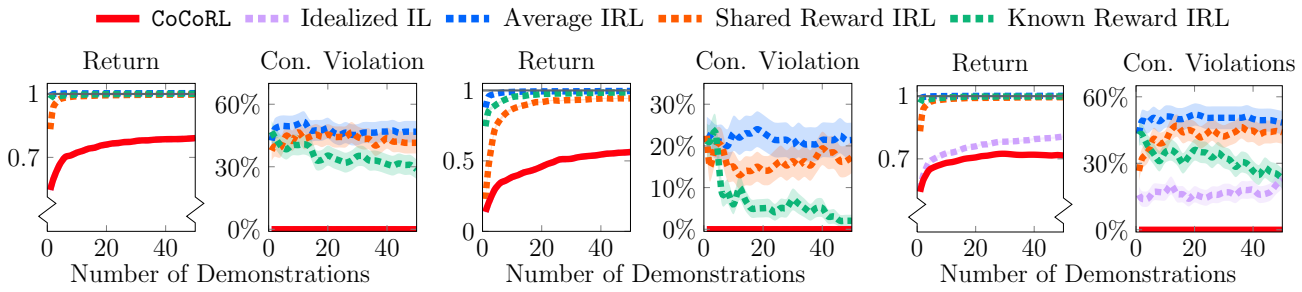
6 Experiments

We present experiments in two domains: (1) tabular environments, where we can solve MDPs and CMDPs exactly (Section 6.2); and (2) a driving simulation, where we investigate the scalability of `CoCoRL` in a more complex environment (Section 6.3). Our experiments assess the safety and performance of `CoCoRL` and compare it to IRL-based baselines (Section 6.1). In Appendix F, we provide more details about the experimental setup, and in Appendix G we present additional experimental results, including an extensive study of `CoCoRL` in one-state CMDPs. We provide code for our experiments at `https://github.com/lasgroup/cocorl`.

6.1 Baselines

To the best of our knowledge, `CoCoRL` is the first algorithm that learns constraints from demonstrations with unknown rewards. Thus it is not immediately clear which baselines to compare it to. Nevertheless, we explore several approaches based on IRL and imitation learining (IL) that could serve as natural starting points for solving the constraint learning problem.

IRL baselines. We consider the following three ways of applying IRL: (1) *Average IRL* infers a separate reward function $\hat{r}_i$ for each demonstration π_i^* and averages the inferred rewards before combining them with an evaluation reward $r_{\text{eval}} + \sum_i \hat{r}_i / k$. (2) *Shared Reward IRL* parameterizes the inferred rewards as $\hat{r}_i + \hat{c}$,



(a) Single environment. (b) Transfer to new task. (c) Transfer to new environment.

Figure 2: Experimental results in Gridworld environments. We consider three settings: (a) no constraint transfer, (b) transferring constraints to new goals in the same grid, and (c) transferring constraints to a new Gridworld with the same structure but different transition dynamics. For each setting, we measure the normalized policy return (**higher is better**), and the constraint violation (**lower is better**). The plots show mean and standard errors over 100 random seeds. `CoCoRL` consistently returns safe solutions, outperforming the IRL-based methods that generally perform worse and are unsafe. The IL baseline performs exactly the same as `CoCoRL` with no environment transfer (the lines overlap in plots a and b), but it produces unsafe solutions with transfer (c). A return greater than 1 indicates a solution that surpasses the best safe policy, implying a constraint violation.

where $\hat{c}$ is shared among all demonstrations. The parameters for both the inferred rewards and the shared constraint penalty are learned simultaneously. (3) *Known Reward IRL* parameterizes the inferred reward as $r_i + \hat{c}$, where r_i is known, and only a shared constraint penalty is learned. Note that *Known Reward IRL* has full access to the demonstrations' rewards that the other methods do not need. Each of these approaches can be implemented with any standard IRL algorithm. In this section, we present results using Maximum Entropy IRL (Ziebart et al., 2008), but in Appendix G we also test Maximum Margin IRL (Ng et al., 2000) and obtain qualitatively similar results.

Idealized IL baseline. Imitation learning (IL) uses supervised learning to imitate a demonstration (Hussein et al., 2017). We can run IL on our set of demonstrations to obtain a set of safe policies. As a baseline, we consider an algorithm that remembers these IL policies and for each evaluation reward r_{eval} chooses the best policy from the IL step. We implement this baseline using an idealized form of imitation learning: we directly use the policies that generated the demonstrations. This gives us an upper bound on the possible performance of a real IL method. For linear rewards and constraints, this algorithm will return the same policies as `CoCoRL`. However it is expensive to run in practice and, as we will see, not robust to changes in the evaluation task or the environment.

For more details about the baselines, see Appendix E.

6.2 Gridworld Environments

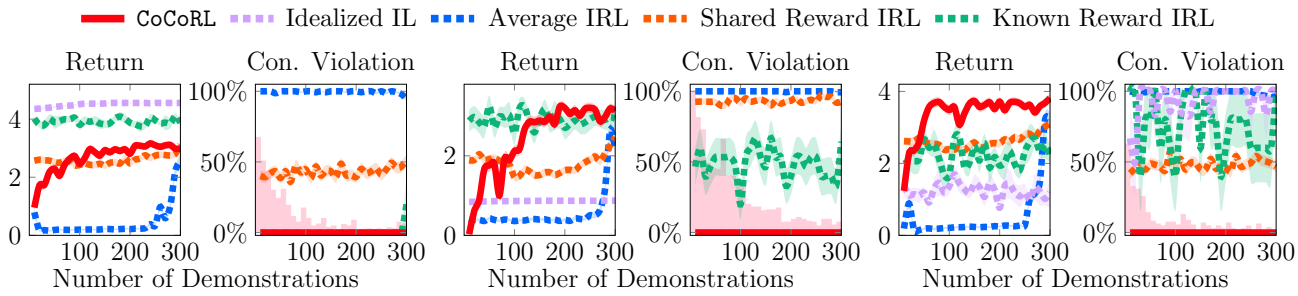
We first consider a set of Gridworld environments which we can, conveniently, solve using linear programming (Altman, 1999). The Gridworlds have goal cells and "limited" cells. When the agent reaches a goal cell, it receives a reward; however, a set of constraints restrict how often the agent can visit the limited cells.

To introduce stochasticity, there is a fixed probability p that the agent executes a random action instead of the intended one. The positions of the goal and limited cells are sampled uniformly at random. The reward and cost values are sampled uniformly from the interval $[0, 1]$. The thresholds are also sampled uniformly, while we ensure feasibility via rejection sampling.

We perform three experiments to evaluate the transfer of learned constraints. First, we evaluate the learned constraints in the same environment and with the same reward distribution that they were learned from. Second, we change the reward distribution by sampling a new set of potential goals. Third, we use a stochastic Gridworld ($p = 0.2$) during training and a deterministic one ($p = 0$) during evaluation.

Figure 2 presents the results of the experiments in 10×10 Gridworlds with 20 goals and 10 limited cells. `CoCoRL` consistently returns safe policies and converges to the best safe solution as more demonstrations are provided. In contrast, none of the IRL-based methods learns safe policies. The IL baseline performs exactly the same as `CoCoRL` without transfer and when transferring to a new task,[4] but it produces unsafe solutions if the environment dynamics change. In the on-distribution evaluation shown in Figure 2a, the IRL methods come closest to the desired safe policy but

[4] Which is expected in tabular environments, as discussed in Appendix E.



(a) Single environment. (b) Transfer to new task. (c) Transfer to new environment.

Figure 3: Return and constraint violation in the `highway-env` intersection environment. All plots show mean and standard error over 5 random seeds with a fixed set of evaluation rewards for each setting. In all settings, `CoCoRL` consistently returns safe policies, as indicated by the low constraint violation values. However, there are instances where `CoCoRL` falls back to providing a default safe solution when the policy optimizer fails to find a feasible solution within the safe set $\mathcal{S}$. The frequency of falling back to the default solution is shown by the bars in the constraint violation plots ( ). In contrast, the IRL method often yields unsafe solutions. IL outperforms `CoCoRL` because we implement an idealized version with perfect imitation. However, for task transfer IL performs much worse than `CoCoRL` and for environment transfer it produces unsafe solutions.

still cause many constraint violations.

6.3 Driving Environment

To explore a more practical and interesting environment, we consider `highway-env`, a 2D driving simulator (Leurent, 2018). We focus on a four-way intersection, as depicted in Figure 1. The agent drives towards the intersection and can turn left, right, or go straight. The environment has a continuous state space, but a discrete high-level action space: the agent can accelerate, decelerate, and make turns at the intersection. There are three possible goals for the agent: turning left, turning right, or going straight. Additionally, the reward functions contain different driving preferences related to velocity and heading angle. Rewards and constraints are defined as linear functions of a state-feature vector that includes goal indicators, vehicle position, velocity, heading, and indicators for unsafe events including crashing and leaving the street.

For constrained policy optimization, we use a constrained cross-entropy method (CEM) based on Wen and Topcu (2018) to optimize a parametric driving controller (see Appendix D for details). We collect synthetic demonstrations that are optimized for different reward functions under a shared set of constraints.

Similar to the previous experiments, we consider three settings: (1) evaluation on the same task distribution and environment; (2) transfer to a new task; and (3) transfer to a modified environment. To transfer to a new task, we learn the constraints from trajectories only involving left and right turns at the intersection, but evaluate them on a reward function that rewards going straight (the situation illustrated in Figure 1). For the transfer to a modified environment, we infer constraints in an environment with very defensive drivers and evaluate them with very aggressive drivers. Appendix F contains additional details on the parameterization of the drivers.

Empirically, we observe that the CEM occasionally fails to find a feasible policy when the safe set $\mathcal{S}$ is small. This is likely due to the CEM relying heavily on random exploration to discover an initial feasible solution. In situations where we cannot find a solution in $\mathcal{S}$, we return a "default" safe controller that remains safe but achieves a return of 0 because it just stands still before the intersection. Importantly, we empirically confirm that whenever we find a solution in $\mathcal{S}$, it is truly safe (i.e., in $\mathcal{F}$).

Figure 3 presents results from the driving experiments. `CoCoRL` consistently guarantees safety, and with a large enough number of demonstrations ($\sim$ 200), `CoCoRL` returns high return policies. In contrast, IRL produces policies with decent performance in terms of return but they tend to be unsafe.

Although the IRL solutions usually avoid crashes (which also result in low rewards), they disregard constraints that are in conflict with the reward function, such as the speed limits and keeping distance to other vehicles. IRL results in substantially more frequent constraint violations when trying to transfer the learned constraint penalty to a new reward. This is likely because the magnitude of the constraint penalty is no longer correct.

The IL baseline performs better than `CoCoRL` when

evaluated in-distribution because our idealized implementation avoid all issues related to optimizing for a policy. However, IL clearly fails in both transfer settings. IL can not learn a policy for a new task that is not in the demonstrations, and it might learn policies via imitation that are no longer safe if the environment dynamics change. CoCoRL does not suffer from this issue, and transferring the inferred constraints to a new reward still results in safe and well-performing policies.

7 Conclusion

We introduced CoCoRL to infer shared constraints from demonstrations with unknown rewards. Theoretical and empirical results show that CoCoRL guarantees safety and achieves strong performance, even when transferring constraints to new tasks or environments.

CoCoRL's main limitation is assuming safe demonstrations and access to a feature representation in non-tabular environments. Learning from potentially unsafe demonstrations and learning features to represent constraints are exciting directions for future work.

Learning constraints has the potential to make RL more sample-efficient and safe. Importantly, CoCoRL can use unlabeled demonstrations, which are often easier to obtain than labelled demonstrations. Thus, CoCoRL can make learning constraints more practical in a range of applications.

Acknowledgements

This project was supported by the Microsoft Swiss Joint Research Center (Swiss JRC), the Swiss National Science Foundation (SNSF) under NCCR Automation, grant agreement 51NF40 180545, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No 815943, the Vienna Science and Technology Fund (WWTF) [10.47379/ICT20058], the Open Philanthropy AI Fellowship, and the Vitalik Buterin PhD Fellowship. We thank Yarden As for valuable discussions throughout the project.

References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2004.

E. Altman. *Constrained Markov decision processes*, volume 7. CRC press, 1999.

K. Amin, N. Jiang, and S. Singh. Repeated inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

M. Babes, V. Marivate, K. Subramanian, and M. L. Littman. Apprenticeship learning about multiple intentions. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.

A. Baddeley, I. Bárány, and R. Schneider. Random polytopes, convex bodies, and approximation. *Stochastic Geometry: Lectures given at the CIME Summer School held in Martina Franca, Italy, September 13–18, 2004*, pages 77–118, 2007.

M. Baert, P. Mazzaglia, S. Leroux, and P. Simoens. Maximum causal entropy inverse constrained reinforcement learning. *arXiv preprint arXiv:2305.02857*, 2023.

C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4): 469–483, 1996.

R. Camilleri, A. Wagenmaker, J. Morgenstern, L. Jain, and K. Jamieson. Active learning with safety constraints. In *Advances in Neural Information Processing Systems*, 2022.

J. Choi and K.-E. Kim. Nonparametric Bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, 2012.

G. Chou, N. Ozay, and D. Berenson. Learning constraints from locally-optimal demonstrations under cost function uncertainty. *IEEE Robotics and Automation Letters*, 5(2):3682–3690, 2020.

C. Dimitrakakis and C. A. Rothkopf. Bayesian multitask inverse reinforcement learning. In *Recent Advances in Reinforcement Learning: 9th European Workshop, EWRL 2011, Athens, Greece, September 9-11, 2011, Revised Selected Papers 9*, pages 273–284. Springer, 2012.

A. Edalat, A. Lieutier, and E. Kashefi. The convex hull in a new model of computation. In *Proceedings of the 13th Canadian Conference on Computational Geometry, University of Waterloo, Ontario, Canada, August 13-15, 2001*, pages 93–96, 2001.

A. Edalat, A. A. Khanban, and A. Lieutier. Computability in computational geometry. In *New Computational Paradigms: First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8-12, 2005. Proceedings 1*, pages 117–127. Springer, 2005.

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

A. Gaurav, K. Rezaee, G. Liu, and P. Poupart. Learning soft constraints from constrained expert demonstrations. In *International Conference on Learning Representations (ICLR)*, 2023.

A. Glazier, A. Loreggia, N. Mattei, T. Rahgooy, F. Rossi, and K. B. Venable. Making human-like trade-offs in constrained environments by learning from demonstrations. *arXiv preprint arXiv:2109.11018*, 2021.

L.-A. Gottlieb, E. Kaufman, A. Kontorovich, and G. Nivasch. Learning convex polytopes with margin. In *Advances in Neural Information Processing Systems*, 2018.

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

S. S. Khan and M. G. Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone. Reward (mis) design for autonomous driving. *Artificial Intelligence*, 316:103829, 2023.

V. Krakovna, J. Uesato, V. Mikulik, M. Rahtz, T. Everitt, R. Kumar, Z. Kenton, J. Leike, and S. Legg. Specification gaming: the flip side of ai ingenuity. *DeepMind Blog*, 2020. https://www.deepmind.com/blog/specification-gaming-the-flip-side-of-ai-ingenuity.

J. Leike, D. Krueger, T. Everitt, M. Martic, V. Maini, and S. Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

E. Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

E. Leurent, Y. Blanco, D. Efimov, and O.-A. Maillard. Approximate robust control of uncertain dynamical systems. In *Advances in Neural Information Processing Systems*, 2018.

D. Lindner, S. Tschiatschek, K. Hofmann, and A. Krause. Interactively learning preference constraints in linear bandits. In *Proceedings of International Conference on Machine Learning (ICML)*, 2022.

G. Liu, Y. Luo, A. Gaurav, K. Rezaee, and P. Poupart. Benchmarking constraint inference in inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

S. Malik, U. Anwar, A. Aghasi, and A. Ahmed. Inverse constrained reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.

P. McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17(2):179–184, 1970.

D. L. McPherson, K. C. Stocking, and S. S. Sastry. Maximum likelihood constraint inference from stochastic demonstrations. In *Conference on Control Technology and Applications (CCTA)*, 2021.

A. Y. Ng, S. Russell, et al. Algorithms for inverse reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2000.

D. Papadimitriou, U. Anwar, and D. S. Brown. Bayesian inverse constrained reinforcement learning. In *Transactions on Machine Learning Research (TMLR)*, 2022.

W. R. Pulleyblank. *Polyhedral combinatorics*. Springer, 1983.

M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

K. Rezaee and P. Yadmellat. How to not drive: Learning driving constraints from demonstration. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022.

R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

D. R. Scobee and S. S. Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

J. Sember. *Guarantees concerning geometric objects with imprecise points*. PhD thesis, University of British Columbia, 2011.

K. C. Stocking, D. L. McPherson, R. P. Matthew, and C. J. Tomlin. Discretizing dynamics for maximum likelihood constraint inference. *arXiv preprint arXiv:2109.04874*, 2021.

P. Wang, H. Li, and C.-Y. Chan. Meta-adversarial inverse reinforcement learning for decision-making tasks. In *International Conference on Robotics and Automation (ICRA)*, 2021.

M. Wen and U. Topcu. Constrained cross-entropy method for safe reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018.

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

L. Yu, T. Yu, C. Finn, and S. Ermon. Meta-inverse reinforcement learning with probabilistic context variables. In *Advances in Neural Information Processing Systems*, 2019.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes**
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes**
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes**

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes**
   (b) Complete proofs of all theoretical results. **Yes**
   (c) Clear explanations of any assumptions. **Yes**

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes**
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes**
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes**
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Yes**
   (b) The license information of the assets, if applicable. **Yes**
   (c) New assets either in the supplemental material or as a URL, if applicable. **Yes**
   (d) Information about consent from data providers/curators. **Not Applicable**
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not Applicable**
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

Appendix

Table of Contents

A Proofs of Theoretical Results

This section provides all proofs omitted in the main paper, as well as a few additional results related to estimating feature expectations (Appendix A.4).

A.1 Limitations of IRL in CMDPs

Proposition 1 (IRL can be unsafe)**.** *There are CMDPs* $\mathcal{C} = (S, A, P, \mu_0, \gamma, r, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$ *such that for any optimal policy* π^* *in* $\mathcal{C}$ *and any reward function* r_{IRL} *that could be returned by an IRL algorithm, the resulting MDP* $(S, A, P, \mu_0, \gamma, r_{IRL})$ *has optimal policies that are unsafe in* $\mathcal{C}$*.*

Proof. This follows from the fact that there are CMDPs for which all feasible policies are stochastic. Every MDP, on the other hand, has a deterministic policy that is optimal (Puterman, 1990).

For example, consider a CMDP with a single state $S = \{s_1\}$ and two actions $A = \{a_1, a_2\}$ with $P(s_1|s_1, a_1) = P(s_1, |s_1, a_2) = 1$. We have two cost functions $c_1(s_1, a_1) = 1, c_1(s_1, a_2) = 0$ and $c_2(s_1, a_1) = 0, c_2(s_1, a_2) = 1$, with thresholds $\xi_1 = \xi_2 = \frac{1}{2}$, and discount factor $\gamma = 0$. To be feasible, a policy needs to have an occupancy measure μ_π such that $\sum_{s,a} \mu(s, a)c_1(s, a) \leq \xi_1$ and $\sum_{s,a} \mu_\pi(s, a)c_2(s, a) \leq \xi_2$.

In our case with a single state and $\gamma = 0$, this simply means that a feasible policy π needs to satisfy $\pi(a_1|s_1) \leq \frac{1}{2}$ and $\pi(a_2|s_1) \leq \frac{1}{2}$. But this implies that only a uniformly random policy is feasible.

Any IRL algorithm infers a reward function that matches the occupancy measure of the expert policy (Abbeel and Ng, 2004). Consequently, the inferred reward r_{IRL} needs to give both actions the same reward, because otherwise π^* would not be optimal for the MDP with r_{IRL}. However, that MDP also has a deterministic optimal policy (like any MDP), which is not feasible in the original CMDP. □

Proposition 2. *Let $\mathcal{C} = (S, A, P, \mu_0, \gamma, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$ be a CMDP without reward. Let r_1, r_2 be two reward functions and π_1^* and π_2^* corresponding optimal policies in $\mathcal{C} \cup \{r_1\}$ and $\mathcal{C} \cup \{r_2\}$. Let $\mathcal{M} = (S, A, P, \mu_0, \gamma)$ be the corresponding MDP without reward. Without additional assumptions, we cannot guarantee the existence of a function $\hat{c} : S \times A \to \mathbb{R}$ such that π_1^* is optimal in the MDP $\mathcal{M} \cup \{r_1 + \hat{c}\}$ and π_2^* is optimal in the MDP $\mathcal{M} \cup \{r_2 + \hat{c}\}$.*

Proof. As in the proof of Proposition 1, we consider a single-state CMDP with $S = \{s_1\}$ and $A = \{a_1, a_2\}$, where a_1. We have $c_1(s_1, a_1) = 1, c_1(s_1, a_2) = 0$ and $c_2(s_1, a_1) = 0, c_2(s_1, a_2) = 1$, $\xi_1 = \xi_2 = \frac{1}{2}$, and $\gamma = 0$. Again, the only feasible policy uniformly randomizes between a_1 and a_2.

Let $r_1(a_1) = 1$, $r_1(a_2) = 1$, $r_2(a_1) = 0$, $r_2(a_2) = 1$, i.e., the first reward function gives equal reward to both actions and the second reward function gives inequal rewards (for simplicity we write $r(a) = r(s, a)$). To make the uniformly random policy optimal in $\mathcal{M} \cup \{r_1 + \hat{c}\}$ and $\mathcal{M} \cup \{r_2 + \hat{c}\}$, both inferred reward functions $r_1 + \hat{c}$ and $r_2 + \hat{c}$ need to give both actions equal rewards. However this is clearly impossible. If $r_1(a_2) + \hat{c}(a_2) - (r_1(a_1) + \hat{c}(a_1)) = 0$, then

$$
\begin{aligned}
& r_2(a_2) + \hat{c}(a_2) - (r_2(a_1) + \hat{c}(a_1)) \\
=& r_2(a_2) - r_1(a_2) + r_1(a_2) + \hat{c}(a_2) - (r_2(a_1) - r_1(a_1) + r_1(a_1) + \hat{c}(a_1)) \\
=& \underbrace{r_2(a_2) - r_1(a_2)}_{=0} - \underbrace{(r_2(a_1) - r_1(a_1))}_{=-1} + \underbrace{r_1(a_2) + \hat{c}(a_2) - (r_1(a_1) + \hat{c}(a_1))}_{=0} = 1 \neq 0.
\end{aligned}
$$

□

A.2 Safety Guarantees

Lemma 1 ($\mathcal{F}$ is convex)**.** *For any CMDP, suppose $\pi_1, \pi_2 \in \mathcal{F} = \{\pi | \forall j : J_j(\pi) \leq \xi_j\}$. Let $\bar{\pi}_{12}$ be a mixture policy such that $\mathbf{f}(\bar{\pi}_{12}) = \lambda \mathbf{f}(\pi_1) + (1 - \lambda)\mathbf{f}(\pi_2)$ with $\lambda \in [0, 1]$. Then, we have $\bar{\pi}_{12} \in \mathcal{F}$.*

Proof. If $\pi_1, \pi_2 \in \mathcal{F}$, then we have $J_j(\pi_1) \leq \xi_j$ and $J_j(\pi_2) \leq \xi_j$ for any j. Further, also for any j, we have $J_j(\bar{\pi}_{12}) = \phi_j^T \mathbf{f}(\bar{\pi}_{12}) = \lambda \phi_j^T \mathbf{f}(\pi_1) + (1 - \lambda)\phi_j^T \mathbf{f}(\pi_2) \leq \xi_j$. Thus, $\bar{\pi}_{12} \in \mathcal{F}$. □

Theorem 1 (Estimated safe set)**.** *Any policy $\pi \in \mathcal{S}$ is safe, i.e., $\pi \in \mathcal{F}$.*

Proof. Our demonstrations $\pi_1^*, \ldots, \pi_k^*$ are all in the true safe set $\mathcal{F}$, and any $\pi \in \mathcal{S}$ is a convex combination of them. Given Lemma 1, we have $\mathcal{S} \subseteq \mathcal{F}$. □

Theorem 2 (Inferred CMDP)**.** *We can find cost functions $\{\hat{c}_j\}_{j=1}^m$ and thresholds $\{\hat{\xi}_j\}_{j=1}^m$ such that for any reward function r_{eval}, solving the inferred CMDP $(S, A, P, \mu_0, \gamma, r_{eval}, \{\hat{c}_j\}_{j=1}^m, \{\hat{\xi}_j\}_{j=1}^m)$ is equivalent to finding $\pi^* \in \text{argmax}_{\pi \in \mathcal{S}} G_{r_{eval}}(\pi)$. Consequently, if $\mathcal{S}$ contains an optimal policy for the true CMDP, solving the inferred CMDP returns a policy that is also optimal in the true CMDP.*

Proof. The convex hull of a set of points is a convex polyhedron, i.e., it is the solution of a set of linear equations (see, e.g., Theorem 2.9 in Pulleyblank, 1983).

Hence, $\mathcal{S}$ is a convex polyhedron in the feature space defined by $\mathbf{f}$, i.e., we can find $A \in \mathbb{R}^{p \times d}, \mathbf{b} \in \mathbb{R}^p$ such that

$$\mathcal{S} = \{x \in \mathbb{R}^d | Ax \leq \mathbf{b}\}.$$

To construct A and $\mathbf{b}$, we need to find the facets of $\mathcal{S}$, the convex hull of a set of points $\mathcal{D}$. This is a standard problem in polyhedral combinatorics (e.g., see Pulleyblank, 1983).

We can now define p linear constraint functions $\hat{c}_j(s,a) = A_j\mathbf{f}(s,a)$ and thresholds $\hat{\xi}_j = b_j$, where A_j is the j-th row of A and b_j is the j-th component of $\mathbf{b}$ (for $j = 1,\dots,p$). Then, solving the CMDP $(S, A, P, \mu_0, \gamma, r_{\text{eval}}, \{\hat{c}_j\}_{j=1}^p \{\hat{\xi}_j\}_{j=1}^p)$ corresponds to solving

$$\pi^* \in \operatorname*{argmax}_{\hat{J}_1(\pi)\le\hat{\xi}_1,\dots,\hat{J}_p(\pi)\le\hat{\xi}_p} G_{r_{\text{eval}}}(\pi).$$

For these linear cost functions, we can rewrite the constraints using the discounted feature expectations as $J_i(\pi) = A_i\mathbf{f}(\pi)$. Hence, π satisfying the constraints $\hat{J}_1(\pi) \le \hat{\xi}_1, \dots, \hat{J}_p(\pi) \le \hat{\xi}_p$ is equivalent to $\pi \in \mathcal{S}$. □

Corollary 1 ($\mathcal{S}$ is maximal)**.** *If $\pi \notin \mathcal{S}$, there exist $r_1,\dots,r_k$ and $c_1,\dots,c_n$, such that the expert policies $\pi_1^*,\dots,\pi_k^*$ are optimal in the CMDPs $(S, A, P, \mu_0, \gamma, r_i, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$, but $\pi \notin \mathcal{F}$.*

Proof. Using Theorem 2, we can construct a set of cost functions $\{c_j\}_{j=1}^n$ and thresholds $\{\xi_j\}_{j=1}^n$ for which $\pi \in \mathcal{S}$ is equivalent to $J_1(\pi) \le \xi_1, \dots, J_n(\pi) \le \xi_n$. Hence, in a CMDP with these cost functions and thresholds, any policy $\pi \notin \mathcal{S}$ is not feasible. Because $\pi_1^*,\dots,\pi_k^* \in \mathcal{S}$ by construction, each π_i^* is optimal in the CMDP $(S, A, P, \mu_0, \gamma, r_i, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$. □

A.3 Optimality Guarantees

Lemma 2. *For any policy π, if $r : S \times A \to [0,1]$ and $\mathbf{f} : S \times A \to [0,1]^d$, we can bound*

$$\forall i: \ \mathbf{f}(\pi)_i \le \frac{1}{1-\gamma} \qquad \|\mathbf{f}(\pi)\|_2 \le \frac{\sqrt{d}}{1-\gamma}$$
$$G_r(\pi) \le \frac{d}{1-\gamma} \qquad \mathcal{R}(\pi,\mathcal{S}) \le \frac{2d}{1-\gamma}$$

Proof. First, we have for any component of the feature expectations

$$\mathbf{f}_i(\pi) = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t\mathbf{f}(s_t,a_t)] \le \sum_{t=0}^{\infty}\gamma^t = \frac{1}{1-\gamma},$$

using the limit of the *geometric series*. This immediately gives

$$\|\mathbf{f}(\pi)\|_2 = \sqrt{\sum_{i=1}^{d}\mathbf{f}(\pi)_i^2} \le \frac{\sqrt{d}}{1-\gamma}.$$

Similarly bounded rewards $r(s,a) \le 1$ imply $\|\theta\|_2 \le \sqrt{d}$. Together, we can bound the returns using the Cauchy-Schwartz inequality

$$G_r(\pi) = \|\theta^T\mathbf{f}(\pi)\|_2 \le \|\theta\|_2 \cdot \|\mathbf{f}(\pi)\|_2 \le \frac{d}{1-\gamma}.$$

Further, for any two policies π_1, π_2, we have $G_r(\pi_1) - G_r(\pi_2) \le \frac{2d}{1-\gamma}$, which implies the same for the regret $\mathcal{R}(r,\mathcal{S}) \le \frac{2d}{1-\gamma}$. □

Theorem 3 (Convergence, exact optimality)**.** *Under Assumptions 1 and 2, for any $\delta > 0$, after $k \ge \log(\delta/f_v(d,n))/\log(1-\delta/f_v(d,n))$, we have $P(\mathcal{R}(r,\mathcal{S}_k) > 0) \le \delta$, where $f_v(d,n)$ is an upper bound on the number of vertices of the true safe set. In particular, $\lim_{k\to\infty}\mathbb{E}_r[\mathcal{R}(r,\mathcal{S}_k)] = 0$.*

Proof. Consider the distribution over optimal policies $P(\mathbf{f}(\pi^*))$ induced by $P(r)$. For each reward r, there is a vertex of the true safe set that is optimal and thus is in the support of $P(\mathbf{f}(\pi^*))$. [5]

[5] Technically, there are degenerate cases where $P(r)$ is only supported on reward functions that are orthogonal to the constraint boundaries and we never see demonstrations at the vertices of the true safe set. This is an artifact of the relatively unnatural assumption of noise-free demonstrations. We can avoid such degenerate cases by mild assumptions: either assuming some minimal noise in $P(r)$ or assuming the algorithm generating the demonstrations has non-zero probability for all policies optimal for a given reward.

We can distinguish two cases. **Case 1**: we have seen a vertex corresponding to an optimal policy for r in the first k demonstrations; and, **Case 2**: we have not. In **Case 1**, we incur 0 regret, and only in **Case 2** we can incur regret greater than 0.

So, we can bound the probability of incurring regret by the probability of **Case 2**:

$$P(\mathcal{R}(r, \mathcal{S}_k) > 0) \leq \sum_{\text{vertex } \mathbf{v}} (1 - P(\mathbf{v}))^k P(\mathbf{v}).$$

To ensure $P(\mathcal{R}(r, \mathcal{S}_k) > 0) \leq \delta$, it is sufficient to ensure that each term of the sum satisfies $(1 - P(\mathbf{v}))^k P(\mathbf{v}) \leq \delta / N_v$ where N_v is the number of vertices. For terms with $P(\mathbf{v}) \leq \delta / N_v$ this is true for all k. For the remaining terms with $P(\mathbf{v}) > \delta / N_v$, we can write

$$(1 - P(\mathbf{v}))^k P(\mathbf{v}) \leq (1 - \delta / N_v)^k.$$

So, it is sufficient to ensure $(1 - \delta / N_v)^k \leq \delta / N_v$, which is satisfied once

$$k > \frac{\log(\delta / N_v)}{\log(1 - \delta / N_v)}.$$

By replacing N_v with a suitable upper bound on the number of vertices, we arrive at the first result.

By Lemma 2, the maximum regret is upper-bounded by $\frac{2d}{1-\gamma}$, and, we can decompose the regret as

$$\mathbb{E}_r[\mathcal{R}(r, \mathcal{S}_k)] \leq \frac{2d}{1-\gamma} \sum_{\text{vertex } \mathbf{v}} (1 - P(\mathbf{v}))^k P(\mathbf{v}) \leq \frac{2d}{1-\gamma} \sum_{\mathbf{v}: P(\mathbf{v}) > 0} (1 - P(\mathbf{v}))^k.$$

Because $P(\mathbf{v})$ is a fixed distribution induced by $P(r)$, the r.h.s. converges to 0 as $k \to \infty$. □

Lemma 3. *Under Assumption 3, we have for any reward function r and any feasible policy $\pi \in \mathcal{F}$*

$$P(\pi) \geq \exp(-\beta d / (1 - \gamma)).$$

Proof. We have $\beta > 0$ and $G_r(\pi) \geq 0$, which implies $\exp(\beta G_r(\pi)) \geq 1$, and

$$P(\pi | r) = \frac{\exp(\beta G_r(\pi)}{Z(\theta)} \geq \frac{1}{Z(\theta)}.$$

By Lemma 2, we have $G_r(\pi) \leq d / (1 - \gamma)$. Therefore,

$$Z(\theta) = \int_{\pi \in \mathcal{F}} \exp(\beta \theta^T \mathbf{f}(\pi)) \mathrm{d}\pi \leq \exp(\beta d / (1 - \gamma)),$$

and

$$P(\pi | r) \geq \exp(-\beta d^2 / (1 - \gamma)).$$

□

Theorem 4 (Convergence, Boltzmann-rationality)**.** *Under Assumption 3, for any $\delta > 0$, after $k \geq \log(\delta / f_v(d, n)) / \log(1 - \exp(-\beta d / (1 - \gamma)))$, we have $P(\mathcal{R}(r, \mathcal{S}_k) > 0) \leq \delta$, where $f_v(d, n)$ is an upper bound on the number of vertices of the true safe set. In particular, $\lim_{k \to \infty} \mathbb{E}_r [\mathcal{R}(r, \mathcal{S}_k)] = 0$.*

Proof. We can upper-bound the probability of having non-zero regret similar to the noise free case by

$$P(\mathcal{R}(r, \mathcal{S}_k) > 0) \leq \sum_{\text{vertex } \mathbf{v}} (1 - P(\mathbf{v}))^k P(\mathbf{v}).$$

Under Assumption 3, we can use Lemma 3, to obtain

$$P(\mathbf{v}) \geq \exp(-\beta d / (1 - \gamma)) \coloneqq \Delta.$$

Importantly, $0 < \Delta < 1$ is a constant, and we have

$$P(\mathcal{R}(r, \mathcal{S}_k) > 0) \leq \sum_{\text{vertex } \mathbf{v}} (1-\Delta)^k.$$

To ensure $P(\mathcal{R}(r, \mathcal{S}_k) > 0) \leq \delta$, it is sufficient to ensure $(1-\Delta)^k \leq \delta / N_v$, where N_v is the number of vertices of the true safe set. This is true once

$$k \geq \frac{\log(\delta / N_v)}{\log(1 - \exp(-\beta d/(1-\gamma)))},$$

where we can again replace N_v by a suitable upper bound.

Bounding the maximum regret via Lemma 2, we get

$$\mathbb{E}_r[\mathcal{R}(r, \mathcal{S}_k)] \leq \frac{2d}{1-\gamma} \sum_{\text{vertex } \mathbf{v}} (1-\Delta)^k,$$

which converges to 0 as $k \to \infty$. □

A.4 Estimating Feature Expectations

Lemma 4. *Let π be a policy with true feature expectation $\mathbf{f}(\pi)$. We estimate the feature expectation using n_{traj} trajectories τ_i collected by rolling out π in the environment: $\hat{\mathbf{f}}(\pi) = \frac{1}{n_{traj}} \sum_{i=1}^{n_{traj}} \mathbf{f}(\tau_i)$. This estimate is unbiased, i.e., $\mathbb{E}[\hat{\mathbf{f}}(\pi)] = \mathbf{f}(\pi)$. Further let $\phi \in \mathbb{R}^d$ be a vector, such that, $0 \leq \phi^T \mathbf{f}(s,a) \leq 1$ for any state s and action a. Then, we have for any $\epsilon > 0$*

$$P(\phi^T \hat{\mathbf{f}}(\pi) - \phi^T \mathbf{f}(\pi) \geq \epsilon) \leq \exp(-2\epsilon^2 n_{traj}(1-\gamma)/d),$$
$$P(\phi^T \mathbf{f}(\pi) - \phi^T \hat{\mathbf{f}}(\pi) \geq \epsilon) \leq \exp(-2\epsilon^2 n_{traj}(1-\gamma)/d).$$

Proof. Because the expectation is linear, the estimate is unbiased, and, $\mathbb{E}[\phi^T \hat{\mathbf{f}}(\pi)] = \phi^T \mathbf{f}(\pi)$.

For any trajectory τ and any $1 \leq i \leq d$, we have $0 \leq \mathbf{f}_i(\tau) \leq 1/(1-\gamma)$, and, consequently, $0 \leq \phi^T \mathbf{f}(\tau) \leq d/(1-\gamma)$, analogously to Lemma 2). Thus, $\phi^T \hat{\mathbf{f}}(\pi)$ satisfies the bounded difference property. In particular, by changing one of the observed trajectories, the value of $\phi^T \hat{\mathbf{f}}(\pi)$ can change at most by $d/(n_{\text{traj}}(1-\gamma))$. Hence, by McDiarmid's inequality, we have for any $\epsilon > 0$

$$P(\theta^T \hat{\mathbf{f}}(\pi) - \mathbb{E}[\theta^T \hat{\mathbf{f}}(\pi)] \geq \epsilon) \leq \exp\left(-\frac{2\epsilon^2}{d/(n_{\text{traj}}(1-\gamma))}\right),$$

and

$$P(\theta^T \hat{\mathbf{f}}(\pi) - \theta^T \mathbf{f}(\pi) \geq \epsilon) \leq \exp\left(-\frac{2\epsilon^2 n_{\text{traj}}(1-\gamma)}{d}\right).$$

We can bound $P(\phi^T \mathbf{f}(\pi) - \phi^T \hat{\mathbf{f}}(\pi) \geq \epsilon)$ analogously. □

Theorem 5 (ϵ-safety with estimated feature expectations.)**.** *Suppose we estimate the feature expectations of each policy π_i^* using at least $n_{traj} > d \log(nk/\delta)/(2\epsilon^2(1-\gamma))$ samples, and construct the estimated safe set $\hat{\mathcal{S}} = \texttt{conv}(\hat{\mathbf{f}}(\pi_1^*), \ldots, \hat{\mathbf{f}}(\pi_k^*))$. Then, we have $P(\max_j (J_j(\pi) - \xi_j) > \epsilon | \pi \in \hat{\mathcal{S}}) < \delta$.*

Proof. Let us define the "good" event that for policy π_i^* we accurately estimate the value of the cost function c_j and denote it by $\mathcal{E}_{ij} = \{\phi_j^T \hat{\mathbf{f}}(\pi_i^*) - \phi_j^T \mathbf{f}(\pi_i^*) \leq \epsilon\}$, where ϕ_j parameterizes c_j. Conditioned on $\mathcal{E}_{ij}$ for all i and j, we have

$$P(\max_j (J_j(\pi) - \xi_j) > \epsilon | \pi \in \hat{\mathcal{S}}, \{\mathcal{E}_{ij}\}) \leq \sum_j P((J_j(\pi) - \xi_j) > \epsilon | \pi \in \hat{\mathcal{S}}, \{\mathcal{E}_{ij}\})$$
$$= \sum_j P(\phi_j^T \sum_l \lambda_l \hat{\mathbf{f}}(\pi_l^*) > \xi_j + \epsilon | \{\mathcal{E}_{ij}\}) = 0.$$

Hence, we can bound the probability of having an unsafe policy by the probability of the "bad" event, which we can bound by Lemma 4 to obtain

$$P(\max_j(J_j(\pi) - \xi_j) > \epsilon | \pi \in \hat{\mathcal{S}}) \leq \sum_{ij} P(\text{not } \mathcal{E}_{ij}) \leq n \cdot k \cdot \exp(-2\epsilon^2 n_{\text{traj}}(1-\gamma)/d).$$

So, to ensure $P(\max_j(J_j(\pi) - \xi_j) > \epsilon | \pi \in \hat{\mathcal{S}}\}) \leq \delta$, it is sufficient to ensure

$$\exp(-2\epsilon^2 n_{\text{traj}}(1-\gamma)/d) \leq \frac{\delta}{nk},$$

which is true if we collect at least

$$n_{\text{traj}} > \frac{d \log(nk/\delta)}{2\epsilon^2(1-\gamma)}$$

trajectories for each policy. □

Theorem 6 (Convergence, noise-free, estimated feature expectations)**.** *Under Assumption 2, if we estimate the feature expectations of at least* $k > \log(\delta/(2f_v(d,n)))/\log(1-\delta/(2f_v(d,n)))$ *expert policies using at least* $n_{traj} > d\log(2f_v(d,n)/\delta)/(2\epsilon^2(1-\gamma))$ *samples each, we have* $P(\mathcal{R}(r,\mathcal{S}_k) > \epsilon) \leq \delta$*, where* $f_v(d,n)$ *is an upper bound on the number of vertices of the true safe set. In particular, we have* $\lim_{\min(k,n_{traj})\to\infty} \mathbb{E}_r[\mathcal{R}(r,\mathcal{S}_k)] = 0$.

Proof. To reason about the possibility of a large estimation error for the feature expectations, we use Lemma 4 to lower bound the probability of the good event $\mathcal{E}_{\mathbf{v}} = \{\theta^T\mathbf{v} - \theta^T\hat{\mathbf{f}}(\pi_i^*) \leq \epsilon\}$, for each vertex $\mathbf{v}$ of the true safe set.

For an evaluation reward r_{eval}, we can incur regret in one of three cases:
Case 1: we have *not* seen a vertex corresponding to an optimal policy for r_{eval}.
Case 2: we have seen an optimal vertex but our estimation of that vertex is *bad*.
Case 3: we have seen an optimal vertex and our estimation of that vertex is *good*.

We show that the probability of **Case 1** shrinks as we increase k, the probability of **Case 2** shrinks as we increase n_{traj}, and the regret that we can incur in **Case 3** is small.

Case 1: This case is independent of the estimated feature expectations. Similar to Theorem 3, we can bound its probability for each vertex $\mathbf{v}$ by $(1-P(\mathbf{v}))^k P(\mathbf{v})$.

Case 2: In case of the bad event, i.e., the complement of $\mathcal{E}_{\mathbf{v}}$, we can upper-bound the probability of incurring high regret using Lemma 4, to obtain

$$P(\theta^T\mathbf{v} - \theta^T\hat{\mathbf{f}}(\pi_i^*) > \epsilon | \text{not } \mathcal{E}_{\mathbf{v}}) P(\text{not } \mathcal{E}_{\mathbf{v}}) \leq P(\text{not } \mathcal{E}_{\mathbf{v}}) \leq \exp(-2\epsilon^2 n_{\text{traj}}(1-\gamma)/d).$$

Case 3: Under the good event $\mathcal{E}_{\mathbf{v}}$, we automatically have regret less than ϵ, i.e.,

$$P(\theta^T\mathbf{v} - \theta^T\hat{\mathbf{f}}(\pi_i^*) > \epsilon | \mathcal{E}_{\mathbf{v}}) P(\mathcal{E}_{\mathbf{v}}) = 0.$$

Using these three cases, we can now bound the probability of having regret greater than ϵ as

$$P(\mathcal{R}(r,\mathcal{S}_k) > \epsilon) \leq \sum_{\text{vertex } \mathbf{v}} \Big(\underbrace{(1-P(\mathbf{v}))^k P(\mathbf{v})}_{\textbf{Case 1}} + \underbrace{\exp(-2\epsilon^2 n_{\text{traj}}(1-\gamma)/d)}_{\textbf{Case 2}} + \underbrace{0}_{\textbf{Case 3}} \Big).$$

To ensure that $P(\mathcal{R}(r,\mathcal{S}_k) > \epsilon) \leq \delta$, it is sufficient to ensure each term of the sum is less than δ/N_v where N_v is the number of vertices. We have two terms inside the sum, so it is sufficient to ensure either term is less than $\delta/(2N_v)$.

Case 1: We argue analogously to Theorem 3. For terms with $P(\mathbf{v}) \leq \delta/(2N_v)$ it is true for all k. For the remaining terms with $P(\mathbf{v}) > \delta/(2N_v)$, we can use

$$(1-P(\mathbf{v}))^k P(\mathbf{v}) \leq (1-\delta/(2N_v))^k$$

so it is sufficient to ensure

$$(1-\delta/(2N_v))^k \leq \delta/(2N_v)$$

which is satisfied once

$$k > \frac{\log(\delta/(2N_v))}{\log(1-\delta/(2N_v))}.$$

This is our first condition.

Case 2: To ensure $\exp(-2\epsilon^2 n_{\text{traj}}(1-\gamma)/d) \leq \delta/(2N_v)$, we need

$$n_{\text{traj}} > \frac{d\log(2N_v/\delta)}{2\epsilon^2(1-\gamma)}.$$

This is our second condition.

If both conditions hold simultaneously, we have $P(\mathcal{R}(r,\mathcal{S}_k) > \epsilon) \leq \delta$.

To analyse the regret, let us consider how ϵ shrinks as a function of n_{traj}. The analysis above holds for any ϵ. So, for a given k and n_{traj}, we need to chose

$$\epsilon^2 \geq \frac{d\log(2nkN_v/\delta)}{2n_{\text{traj}}(1-\gamma)}$$

to guarantee $P(\mathcal{R}(r,\mathcal{S}_k) > \epsilon) \leq \delta$. Hence, the smallest ϵ we can choose is

$$\epsilon_{\min} = \sqrt{\frac{d\log(2nkN_v/\delta)}{2n_{\text{traj}}(1-\gamma)}}.$$

Using Lemma 2 to upper-bound the maximum regret by $\frac{2d^2}{1-\gamma}$, we can upper-bound the expected regret by

$$\begin{aligned}
&\mathbb{E}_r[\mathcal{R}(r,\mathcal{S}_k)] \leq \frac{2d^2}{1-\gamma}P(\mathcal{R}(r,\mathcal{S}_k) > \epsilon_{\min}) + \epsilon_{\min}P(\mathcal{R}(r,\mathcal{S}_k) \leq \epsilon_{\min}) \\
&\leq \frac{2d}{1-\gamma}\sum_{\mathbf{v}:P(\mathbf{v})>0}(1-P(\mathbf{v}))^k + \frac{2dN_v}{1-\gamma}\exp(-2\epsilon_{\min}^2(1-\gamma)/d)\frac{k}{\exp(n_{\text{traj}})} + \sqrt{\frac{d\log(2nkN_v/\delta)}{2n_{\text{traj}}(1-\gamma)}}.
\end{aligned}$$

For each term individually, we can see that it converges to 0 as $\min(k, n_{\text{traj}}) \to \infty$:

$$\lim_{\min(k,n_{\text{traj}})\to\infty}\sum_{\mathbf{v}:P(\mathbf{v})>0}(1-P(\mathbf{v}))^k = \lim_{k\to\infty}\sum_{\mathbf{v}:P(\mathbf{v})>0}(1-P(\mathbf{v}))^k = 0,$$

$$\lim_{\min(k,n_{\text{traj}})\to\infty}\underbrace{\exp(-2\epsilon_{\min}^2(1-\gamma)/d)}_{\leq 1}\frac{k}{\exp(n_{\text{traj}})} \leq \lim_{\min(k,n_{\text{traj}})\to\infty}\frac{k}{\exp(n_{\text{traj}})} = 0,$$

$$\lim_{\min(k,n_{\text{traj}})\to\infty}\sqrt{\frac{d\log(2nkN_v/\delta)}{2n_{\text{traj}}(1-\gamma)}} = 0.$$

Hence, $\lim_{\min(k,n_{\text{traj}})\to\infty}\mathbb{E}_r[\mathcal{R}(r,\mathcal{S}_k)] = 0$. □

Theorem 7 (Convergence under Boltzmann noise, estimated feature expectations)**.** *Under Assumption 3, if we estimate the feature expectations of at least* $k > \log(\delta/(2f_v(d,n)))/(\log(1-\exp(-\beta d/(1-\gamma))))$ *expert policies using at least* $n_{traj} > d\log(2f_v(d,n)/\delta)/(2\epsilon^2(1-\gamma))$ *samples each, we have* $P(\mathcal{R}(r,\mathcal{S}_k) > \epsilon) \leq \delta$*, where* $f_v(d,n)$ *is an upper bound on the number of vertices of the true safe set. In particular, we have* $\lim_{\min(k,n_{traj})\to\infty}\mathbb{E}_r[\mathcal{R}(r,\mathcal{S}_k)] = 0$.

Proof. We can decompose the probability of incurring regret greater than ϵ into the same three cases discussed in the proof of Theorem 6 to get the upper-bound

$$P(\mathcal{R}(r, \mathcal{S}_k) > \epsilon) \leq \sum_{\text{vertex } \mathbf{v}} \Big(\underbrace{(1 - P(\mathbf{v}))^k P(\mathbf{v})}_{\textbf{Case 1}} + \underbrace{\exp(-2\epsilon^2 n_{\text{traj}}(1-\gamma)/d)}_{\textbf{Case 2}} + \underbrace{0}_{\textbf{Case 3}} . \Big)$$

Under Assumption 3, we can use Lemma 3, to get

$$P(\mathbf{v}) \geq \exp(-\beta d/(1-\gamma)) \coloneqq \Delta.$$

Combining both bounds, we get

$$P(\mathcal{R}(r, \mathcal{S}_k) > \epsilon) \leq \sum_{\text{vertex } \mathbf{v}} \Big((1-\Delta)^k + \exp(-2\epsilon^2 n_{\text{traj}}(1-\gamma)/d) \Big).$$

To ensure $P(\mathcal{R}(r, \mathcal{S}_k) > \epsilon) \leq \delta$ it is sufficient to ensure both $(1-\Delta)^k \leq \delta/(2N_v)$ and $\exp(-2\epsilon^2 n_{\text{traj}}(1-\gamma)/d \leq \delta/(2N_v)$ where N_v is the number of vertices of the true safe set. This is true once

$$k > \frac{\log(\delta/(2N_v))}{\log(1-\Delta)} = \frac{\log(\delta/(2N_v))}{\log(1 - \exp(-\beta d/(1-\gamma)))},$$

and

$$n_{\text{traj}} > \frac{d \log(2N_v/\delta)}{2\epsilon^2(1-\gamma)},$$

where can again replace N_v by a suitable upper bound.

For the regret, we get asymptotic optimality with the same argument from the proof of Theorem 6, replacing $P(\mathbf{v})$ with the constant Δ. □

B Constructing a Conservative Safe Set from Estimated Feature Expectations

Theorem 5 shows that we can guarantee ϵ-safety even when estimating the feature expectations of the demonstrations. However, sometimes we need to ensure exact safety. In this section, we discuss how we can use confidence intervals around estimated feature expectations to construct a conservative safe set, i.e., a set $\hat{\mathcal{S}} \subset \mathcal{S}$ that still guarantees *exact* safety with high probability (w.h.p.).

Suppose we have k confidence sets $\mathcal{C}_i \subseteq \mathbb{R}^d$, such that we know $\mathbf{f}(\pi_i^*) \in \mathcal{C}_i$ (w.h.p.). Now we want to construct a conservative convex hull, i.e., a set of points in which any point certainly in the convex hull of $\mathbf{f}(\pi_1^*), \dots, \mathbf{f}(\pi_k^*)$. In other words, we want to construct the intersection of all possible convex hulls with points from $\mathcal{C}_1, \dots, \mathcal{C}_k$:

$$\hat{S} = \bigcap_{x_1, \dots, x_k \in \mathcal{C}_1 \times \cdots \times \mathcal{C}_k} \texttt{conv}(x_1, \dots, x_k).$$

This set is sometimes called the *guaranteed hull* (Sember, 2011), or the *interior convex hull* (Edalat et al., 2005). Efficient algorithms are known to construct guaranteed hulls in 2 or 3 dimensions if $\mathcal{C}$ has a simple shape, such as a cube or a disk (e.g., see Sember, 2011). However, we need to construct a guaranteed hull in d dimensions.

Edalat et al. (2001) propose to reduce the problem of constructing guaranteed hull when $\mathcal{C}_i$ are (hyper-)rectangles to computing the intersection of convex hulls constructed from combinations of the corners of the rectangle. In $d \leq 3$ dimensions, this is possible with $\mathcal{O}(k \log k)$ complexity, but in higher dimensions is requires $\mathcal{O}(k^{\lfloor d/2 \rfloor})$. If we use a concentration inequality for each coordinate independently, e.g., by using Lemma 4 with unit basis vectors, then the $\mathcal{C}_i$'s are rectangles, and we can use this approach. However, because we have to construct many convex hulls, it can get expensive as k and d grow.

Figure 4 illustrates the guaranteed hull growing as the confidence regions shrink. The guaranteed hull is a conservative estimate and can be empty if the $\mathcal{C}_i$'s are too large. Hence, $\hat{\mathcal{S}}$ might be too conservative sometimes, especially if we cannot observe multiple trajectories from the same policy. In such cases, we need additional assumptions about the environment (e.g., it is deterministic) or the demonstrations (e.g., every trajectory is safe) to guarantee safety in practice. Still, using a guaranteed hull as a conservative safe set is a promising alternative to relying on point estimates of the feature expectations.

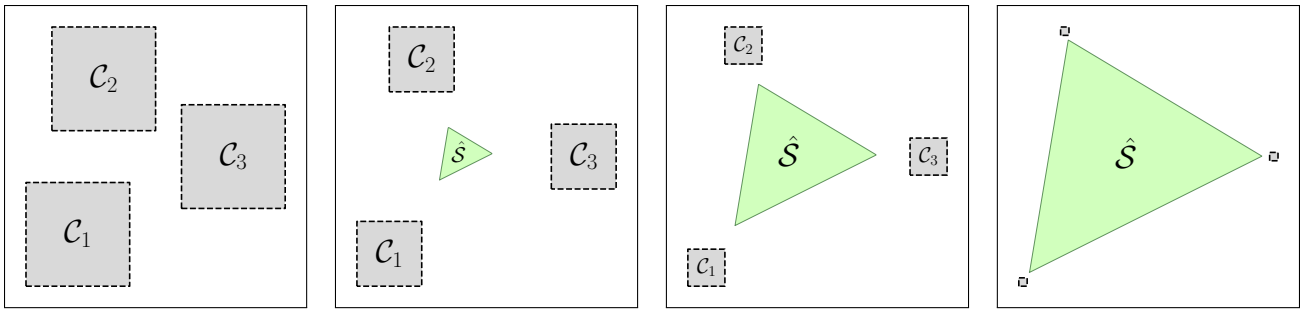


Figure 4: Illustration of the *guaranteed hull* in 2D. The gray areas depict confidence sets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$, and the green area is the conservative safe set $\hat{\mathcal{S}}$ constructed using the guaranteed hull. For large confidence sets, the guaranteed hull can be empty (left); for small confidence sets, it approaches the safe set constructed using the exact feature expectations (right).

C Excluding Guaranteed Unsafe Policies

In the main paper, we showed that $\mathcal{S}$ guarantees safety and that we cannot safely expand that set. In this section, we discuss how we can distinguish policies that we know are unsafe from policies we are still uncertain about. First, we note that, in general, this is impossible. For example, if the expert reward functions are all-zero $r_i(s,a) = 0$, any policy outside $\mathcal{S}$ might still be in $\mathcal{F}$. However, if we exclude such degenerate cases, we can expect expert policies to lie on the boundary of $\mathcal{F}$, which gives us some additional information.

Hence, in this section, we assume optimal demonstrations (Assumption 2), and we make an additional (mild) assumption to exclude degenerate demonstrations.

Assumption 4 (Non-degenerate demonstrations)**.** For each expert reward function r_i there are at least two expert policies π_j^*, π_l^* that have different returns under r_i, i.e., $G_{r_i}(\pi_j^*) \neq G_{r_i}(\pi_l^*)$.

Now, we construct the "unsafe" set $\mathcal{U} = \cup_{i=1}^k \mathcal{U}_i$ with

$$\mathcal{U}_i = \{x \in \mathbb{R}^d | \exists \alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots \alpha_k > 0 \text{ s.t. } x - \mathbf{f}(\pi_i^*) = \sum_{j \neq i} \alpha_j (\mathbf{f}(\pi_i^*) - \mathbf{f}(\pi_j^*))\}.$$

For simplicity, we will sometimes write $\pi \in \mathcal{U}$ instead of $\mathbf{f}(\pi) \in \mathcal{U}$. At each vertex i of the safe set, $\mathcal{U}_i$ is a convex cone of points incompatible with observing π_i^*. In particular, any point in $\mathcal{U}_i$ is strictly better than π_i^*, i.e., for any reward function for which π_i^* is optimal, a point in $\mathcal{U}_i$ would be better than π_i^*. Figure 5 illustrates this construction. We can show that any policy in $\mathcal{U}$ is guaranteed not to be in $\mathcal{F}$, and that $\mathcal{U}$ is the largest possible set with this property.

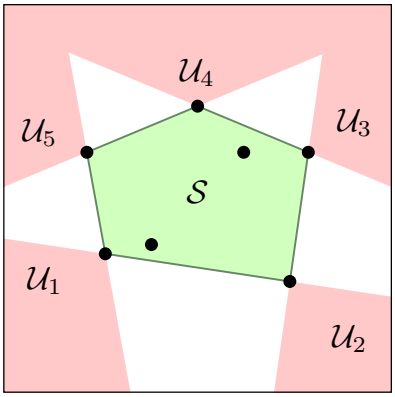


Figure 5: Illustration of safe set and "unsafe" set in 2D.

Theorem 8. *Under Assumption 2 and Assumption 4, $\mathcal{U} \cap \mathcal{F} = \emptyset$. In particular, any $\pi \in \mathcal{U}$ is not in $\mathcal{F}$.*

Proof. Assume there is a policy $\pi \in \mathcal{U} \cap \mathcal{F}$. Then, by construction, there is an $1 \leq i \leq k$ such that $\pi \in \mathcal{U}_i$, i.e., there are $\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_k > 0$, such that

$$\mathbf{f}(\pi) - \mathbf{f}(\pi_i^*) = \sum_{j \neq i} \alpha_j (\mathbf{f}(\pi_i^*) - \mathbf{f}(\pi_j^*)).$$

We know that π_i^* is optimal for a reward function $r_i(s,a) = \theta_i^T \mathbf{f}(s,a)$ parameterized by $\theta_i \in \mathbb{R}^d$. So, for any other demonstration π_j^*, we necessarily have $\theta_i^T(\mathbf{f}(\pi_i^*) - \mathbf{f}(\pi_j^*)) \geq 0$.

Because $\pi \in \mathcal{U}_i$, we have

$$\theta_i^T(\mathbf{f}(\pi) - \mathbf{f}(\pi_i^*)) = \sum_{j \neq i} \alpha_j \theta_i^T(\mathbf{f}(\pi_i^*) - \mathbf{f}(\pi_j^*)) \geq 0.$$

However, by Assumption 4, at least one j has $G_{r_i}(\pi_i^*) > G_{r_i}(\pi_j^*)$. And, because we sum over all j with all $\alpha_j > 0$, this implies $\theta_i^T \mathbf{f}(\pi) > \theta_i^T \mathbf{f}(\pi_i^*)$. However, this is a contradiction with π_i^* being optimal for reward r_i within $\mathcal{F}$. Hence, $\mathcal{U} \cap \mathcal{F} = \emptyset$. □

Theorem 9 (Set $\mathcal{U}$ is maximal)**.** *Under Assumption 2 and Assumption 4, if $\pi \notin \mathcal{U}$, there are reward functions $r_1, \dots, r_k$, cost functions $c_1, \dots, c_n$, and thresholds $\xi_1, \dots, \xi_n$, such that, each policy π_i^* is optimal in the CMDP $(S, A, P, \mu_0, \gamma, r_i, \{c_j\}_{j=1}^n, \{\xi_j\}_{j=1}^n)$, and π is feasible, i.e., $\pi \in \mathcal{F}$.*

Proof. If $\pi \in \mathcal{S}$, the statement follows from Theorem 2. So, it remains to show the statement holds when $\pi \notin \mathcal{S}$ and $\pi \notin \mathcal{U}$. Consider a CMDP with the true safe set

$$\mathcal{F} = \{\pi' | \mathbf{f}(\pi') \in \texttt{conv}(\mathbf{f}(\pi_1^*), \dots, \mathbf{f}(\pi_k^*), \mathbf{f}(\pi))\}.$$

As this is a convex polyhedron, we can write it in terms of linear equations (similar to Theorem 2); so, there is a CMDP with this true safe set.

We can prove the result, by finding reward functions $r_1, \dots, r_k$ such that the expert policies $\pi_i^* \in \operatorname{argmax}_{\pi \in \mathcal{F}} G_{r_i}(\pi)$ are optimal, but $\pi \notin \operatorname{argmax}_{\pi \in \mathcal{F}} G_{r_i}(\pi)$ is not optimal for any of the rewards. With this choice, we will observe the same expert policies $\pi_1^*, \dots, \pi_k^*$; but the true safe set has an additional vertex π, which we never observe.

Now, we show that it is possible for any $\pi \notin \mathcal{U}$. This means, we cannot expand $\mathcal{U}$ while guaranteeing that all policies in it are unsafe. The key is to show that all points $\mathbf{f}(\pi_1^*), \dots, \mathbf{f}(\pi_k^*), \mathbf{f}(\pi)$ are vertices of $\mathcal{F}$, which is the case only if $\pi \notin \mathcal{U}$.

We can assume w.l.o.g that the k demonstrations are vertices of $\texttt{conv}(\mathbf{f}(\pi_1^*), \dots, \mathbf{f}(\pi_k^*))$. So we only need to show that no $\mathbf{f}(\pi_i^*)$ is a convex combination of $\{\mathbf{f}(\pi_j^*)\}_{j \neq i} \cup \{\mathbf{f}(\pi)\}$. Assume this was true, i.e., there are $\alpha_j > 0$, $\alpha > 0$ with $\alpha + \sum_{j \neq i} \alpha_j = 1$ and

$$\mathbf{f}(\pi_i^*) = \alpha \mathbf{f}(\pi) + \sum_{j \neq i} \alpha_j \mathbf{f}(\pi_j^*).$$

Then, we could rewrite

$$\begin{aligned}
\mathbf{f}(\pi) &= \frac{1}{\alpha}\mathbf{f}(\pi_i^*) - \sum_{j \neq i} \frac{\alpha_j}{\alpha}\mathbf{f}(\pi_j^*) \\
&= \frac{1}{\alpha}\mathbf{f}(\pi_i^*) - \frac{1-\alpha}{\alpha}\mathbf{f}(\pi_i^*) + \frac{1-\alpha}{\alpha}\mathbf{f}(\pi_i^*) - \sum_{j \neq i} \frac{\alpha_j}{\alpha}\mathbf{f}(\pi_j^*) \\
&= \mathbf{f}(\pi_i^*) + \sum_{j \neq i} \frac{\alpha_j}{\alpha}(\mathbf{f}(\pi_i^*) - \mathbf{f}(\pi_j^*)).
\end{aligned}$$

This implies that $\pi \in \mathcal{U}_i$ because $\alpha_j / \alpha > 0$ for all j. However, this is a contradiction to $\pi \notin \mathcal{U}$; thus, $\mathbf{f}(\pi_1^*), \dots, \mathbf{f}(\pi_k^*), \mathbf{f}(\pi)$ are vertices of $\mathcal{F}$.

Given that the optimal policies are vertices of $\mathcal{F}$, we can choose k linear reward functions as follows:

1. For i, let $A_i = \texttt{conv}(\mathbf{f}(\pi_1^*), \dots, \mathbf{f}(\pi_{i-1}^*), \mathbf{f}(\pi_{i+1}^*), \dots, \mathbf{f}(\pi_k^*), \mathbf{f}(\pi))$ and $B_i = \{\mathbf{f}(\pi_i^*)\}$.
2. A_i and B_i are disjoint convex sets and by the separating hyperplane theorem, we can find a hyperplane that separates them.
3. Choose θ_i as the orthogonal vector to the separating hyperplane, oriented from A_i to B_i.

With this construction π_i^* is optimal only for $r_i(s, a) = \theta_i \mathbf{f}(s, a)$ and π is optimal for none of the reward functions. Therefore, any $\pi \notin \mathcal{U}$ could be an additional vertex of the true safe set, that we did not observe in the demonstrations. In other words, we cannot expand $\mathcal{U}$. □

Using this construction we can separate the feature space into policies we know are safe ($\pi \in \mathcal{S}$), policies we know are unsafe ($\pi \in \mathcal{U}$), and policies we are uncertain about ($\pi \notin \mathcal{S} \cup \mathcal{U}$). This could be useful to measure the ambiguity of a set of demonstrations. The policies we are uncertain about could be a starting point for implementing an *active learning* version of `CoCoRL`.

Algorithm 1 Convex Constraint Learning for Reinforcement Learning

function `constraint_learning`($\mathcal{D}$, n_{points}, d_{stop})
  $(i, d_{\text{next}}) \leftarrow (0, \infty)$
  $x_{\text{next}} \leftarrow$ random starting point from $\mathcal{D}$
  while $i \leq n_{\text{points}}$ and $|\mathcal{D}| > 0$ and $d_{\text{next}} > d_{\text{stop}}$ **do**
    $\bar{\mathcal{D}} \leftarrow \bar{\mathcal{D}} \cup \{x_{\text{next}}\}$, $\mathcal{D} \leftarrow \mathcal{D} \setminus \{x_{\text{next}}\}$, $i \leftarrow i + 1$
    $\mathcal{S} \leftarrow$ `convex_hull`($\bar{\mathcal{D}}$)
    $(x_{\text{next}}, d_{\text{next}}) \leftarrow$ `furthest_point`($\mathcal{D}, \mathcal{S}$)
  $\mathcal{U} \leftarrow$ `unsafe_set`($\mathcal{S}$)
  return $\mathcal{S}, \mathcal{U}$

function `furthest_point`($\mathcal{X}, \mathcal{P}$)
  $d_{\max} = -\infty$
  for $x \in \mathcal{X}$ **do**
    Solve QP to find $d \in \text{argmin}_{p \in \mathcal{P}} (x - p)^2$
    if $d > d_{\max}$ **then** $d_{\max} \leftarrow d$, $x_{\max} \leftarrow x$
  return $(x_{\max}, d_{\max})$

function convex_hull($\mathcal{X}$)
  if If $|\mathcal{X}| < 3$ **then return** special case solution $A, \mathbf{b}$
  Determine effective dimension of $\mathcal{X}$
  $\mathcal{X}_{\text{proj}} \leftarrow$ project $\mathcal{X}$ to lower dimensional space
  $H_{\text{proj}} \leftarrow$ call Qhull to obtain convex hull of $\mathcal{X}_{\text{proj}}$
  $A_{\text{proj}}, \mathbf{b}_{\text{proj}} \leftarrow$ call Qhull to obtain linear equations describing H_{proj}
  $A, \mathbf{b} \leftarrow$ project $A_{\text{proj}}, \mathbf{b}_{\text{proj}}$ back to $\mathbb{R}^d$
  return $A, \mathbf{b}$

D `CoCoRL` Implementation Details

This section discusses a few details of our implementation of `CoCoRL`. In particular, we highlight practical modifications related to constructing the convex hull $\mathcal{S}$: how we handle degenerate sets of demonstrations and how we greedily select which points to use to construct the convex hull. Algorithm 1 shows full pseudocode for our implementation of `CoCoRL`.

D.1 Constructing the Convex Hull

We use the Quickhull algorithm (Barber et al., 1996) to construct convex hulls. In particular, we use a standard implementation, called *Qhull*, which can construct the convex hull from a set of points and determine the linear equations. However, it assumes at least 3 input points, and non-degenerate inputs, i.e., it assumes that $\text{rank}(\mathbf{f}(\pi_1^*), \dots, \mathbf{f}(\pi_k^*)) = d$. To avoid these limiting cases, we make two practical modifications.

Special-case solutions for 1 and 2 inputs. For less than 3 input points it is easy to construct the linear equations describing the convex hull manually. If we only see a single demonstration π_1^*, the convex hull is simply, $\texttt{conv}(\mathbf{f}(\pi_1^*)) = \{\mathbf{f}(\pi_1^*)\} = \{x | Ax \leq \mathbf{b}\}$ with

$$A = [I_d, -I_d]^T \qquad \mathbf{b} = [\mathbf{f}(\pi_1^*)^T, \mathbf{f}(\pi_1^*)^T]^T.$$

If we observe 2 demonstrations π_1^*, π_2^*, the convex hull is given by

$$\texttt{conv}(\mathbf{f}(\pi_1^*), \mathbf{f}(\pi_2^*)) = \{\mathbf{f}(\pi_1^*) + \lambda(\mathbf{f}(\pi_2^*) - \mathbf{f}(\pi_1^*)) | 0 \leq \lambda \leq 1\} = \{x | Ax \leq \mathbf{b}\},$$

where

$$A = [W, -W, \mathbf{v}^T, -\mathbf{v}^T]^T \qquad \mathbf{b} = [W\mathbf{f}(\pi_1^*), -W\mathbf{f}(\pi_1^*), \mathbf{v}^T\mathbf{f}(\pi_2^*), -\mathbf{v}^T\mathbf{f}(\pi_1^*)]^T.$$

Here, $\mathbf{v} = \mathbf{f}(\pi_2^*) - \mathbf{f}(\pi_1^*)$, and $W = [\mathbf{w}_1, \dots, \mathbf{w}_{d-1}]$ spans the space orthogonal to $\mathbf{v}$, i.e., $\mathbf{w}_i^T\mathbf{v} = 0$.

Handling degenerate safe sets. If $\text{rank}(\mathbf{f}(\pi_1^*), \dots, \mathbf{f}(\pi_k^*)) < d$, we first project the demonstrations to a lower-dimensional subspace in which they are full rank, then we call Qhull to construct the convex hull in this space. Finally, we project back the linear equations describing the convex hull to $\mathbb{R}^d$. To determine the correct projection, let us define the matrix $D = [\mathbf{f}(\pi_1^*)^T, \dots, \mathbf{f}(\pi_k^*)^T]^T \in \mathbb{R}^{k \times d}$. Now, we can do the singular value decomposition (SVD): $D = U^T \Sigma V$, where $U = [\mathbf{u}_1, \dots, \mathbf{u}_k] \in \mathbb{R}^{k \times k}$, $V = [\mathbf{v}_1, \dots, \mathbf{v}_d] \in \mathbb{R}^{d \times d}$, and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_d) \in \mathbb{R}^{k \times d}$. Assuming the singular values are ordered in decreasing order by magnitude., we can determine the effective dimension $\tilde{d}$ of the demonstrations such that $\sigma_{\tilde{d}} > 0$ and $\sigma_{\tilde{d}+1} = 0$ (for numerical stability, we use a small positive number instead of 0). Now, we can define projection matrices

$$X_{\text{ef}} = [\mathbf{u}_1, \dots, \mathbf{u}_{\tilde{d}}]^T \in \mathbb{R}^{\tilde{d} \times d},$$
$$X_{\text{orth}} = [\mathbf{u}_{\tilde{d}+1}, \dots, \mathbf{u}_d]^T \in \mathbb{R}^{(d-\tilde{d}) \times d},$$

where X_{ef} projects to the span of the demonstrations, and X_{orth} projects to the complement. Using X_{ef}, we can project the demonstrations to their span and construct the convex hull in that projection, $\texttt{conv}(X_{\text{ef}} D) = \{x | A_{\text{proj}} x \leq \mathbf{b}_{\text{proj}}\}$. To project the convex hull back to $\mathbb{R}^d$, we construct linear equations in $\mathbb{R}^d$, such that $\texttt{conv}(D) = \{x | Ax \leq \mathbf{b}\}$:

$$A = [(X_{\text{ef}}^T A_{\text{proj}})^T, X_{\text{orth}}, -X_{\text{orth}}]^T \qquad \mathbf{b} = [\mathbf{b}_{\text{proj}}, X_{\text{orth}} \mathbf{f}(\pi_1^*), -X_{\text{orth}} \mathbf{f}(\pi_1^*)],$$

where $X_{\text{ef}}^T A_{\text{proj}}$ projects the linear equations back to $\mathbb{R}^d$, and the other two components ensure that $X_{\text{orth}} x = X_{\text{orth}} \mathbf{f}(\pi_1^*)$, i.e., restricts the orthogonal components to the convex hull. In practice, we change this constraint to $-X_{\text{orth}} \mathbf{f}(\pi_1^*) - \epsilon \leq X_{\text{orth}} x \leq X_{\text{orth}} \mathbf{f}(\pi_1^*) + \epsilon$ for some small $\epsilon > 0$.

D.2 Iteratively Adding Points

Constructing the safe set from all demonstrations can sometimes be problematic, especially if it results in too many inferred constraints. This scenario is more likely to occur when demonstrations are clustered closely together in feature space, which occurs, e.g., if they are not exactly optimal. To mitigate such problems, we adopt an iterative approach for adding points to the safe set. We start with a random point from the set of demonstrations, and subsequently, we iteratively add the point farthest away from the existing safe set. We can determine the distance of a given demonstration π by solving the quadratic program

$$d \in \underset{x \in \mathcal{S}}{\text{argmin}} (x - \mathbf{f}(\pi))^2.$$

We solve this problem for each remaining demonstration from the safe set to determine which point to add next. We stop adding points once the distance of the next point we would add gets too small. By changing the stopping distance as a hyperparameter, we can trade-off between adding all points important for expanding the safe set and regularizing the safe set.

E Details on the Baselines

In this section, we discuss the adapted IRL and IL baselines we present in the main paper in more detail.

E.1 IRL Baselines

In this section, we discuss how we adapt maximum margin IRL and maximum entropy IRL to implement *Average IRL*, *Shared Reward IRL*, and *Known Reward IRL*.

Recall that these three variants model the expert reward function as one of:

$$\hat{r}_i(s,a) \qquad \text{(Average IRL)}$$
$$\hat{r}_i(s,a) = r_i^{\text{known}}(s,a) + \hat{c}(s,a) \qquad \text{(Known Reward IRL)}$$
$$\hat{r}_i(s,a) = \hat{r}_i(s,a) + \hat{c}(s,a) \qquad \text{(Shared Reward IRL)}$$

E.1.1 Maximum-Margin IRL

Let us first recall the standard maximum-margin IRL problem (Ng et al., 2000). We want to infer a reward function $\hat{r}$ from an expert policy π_E, such that the expert policy is optimal under the inferred reward function. To be optimal is equivalent to $\hat{V}^{\pi_E}(s) \geq \hat{Q}^{\pi_E}(s,a)$ for all states s and actions a. Here $\hat{V}^{\pi_E}$ is the value function of the expert policy computed under reward $\hat{r}$ and $\hat{Q}^{\pi_E}(s,a)$ is the corresponding Q-function.

However, this condition is underspecified. In maximum margin IRL, we resolve the ambiguity between possible solutions by looking for the inferred reward function $\hat{r}$ that maximizes $\sum_{s,a}(\hat{V}^{\pi_E}(s) - \hat{Q}^{\pi_E}(s,a))$, i.e., the *margin* by which π_E is optimal.

The goal of maximum margin IRL is to solve the optimization problem

$$\begin{aligned} \underset{\hat{r},\zeta}{\text{maximize}} \quad & \sum_i \zeta_i \\ \text{subject to} \quad & \hat{V}^{\pi_E}(s) - \hat{Q}^{\pi_E}(s,a) \geq \zeta, \quad \forall (s,a) \in S \times A \\ & -1 \leq \hat{r}(s,a) \leq 1 \end{aligned} \tag{1}$$

In a tabular environment, we can write this as a linear program because both $\hat{V}^{\pi_E}$ and $\hat{Q}^{\pi_E}$ are linear in the state-action occupancy vectors.

For simplicity, let us consider only state-dependent reward functions $r(s,a,s') = r(s')$. Also, let us introduce the vector notation

$$\begin{aligned} \mathbf{r} &\coloneqq (r(s_1), r(s_2), \dots)^T \in \mathbb{R}^{|S|} \\ \mathbf{V}^\pi &\coloneqq (V(s_1), V(s_2), \dots)^T \in \mathbb{R}^{|S|} \\ \mathbf{Q}_a^\pi &\coloneqq (Q(s_1,a), Q(s_2,a), \dots)^T \in \mathbb{R}^{|S|} \end{aligned}$$

$$P^\pi \coloneqq \begin{pmatrix} \sum_a P(s_1|s_1,a)\pi(a|s_1) & \sum_a P(s_1|s_2,a)\pi(a|s_2) & \dots \\ \sum_a P(s_2|s_1,a)\pi(a|s_1) & \ddots & \dots \\ \vdots & \dots & \dots \end{pmatrix} \in \mathbb{R}^{|S|\times|S|}$$

$$P^a \coloneqq P^{\pi_a} \text{ with } \pi_a(a|s) = 1$$

Now, we can write Bellman-like identities in this vector notation:

$$\begin{aligned} \mathbf{V}^\pi &= P^\pi(\mathbf{r} + \gamma \mathbf{V}^\pi) \\ \mathbf{V}^\pi &= (I - \gamma P^\pi)^{-1} P^\pi \mathbf{r} \\ \mathbf{Q}_a^\pi &= P^a(\mathbf{r} + \gamma \mathbf{V}^\pi) \end{aligned}$$

We can also write the maximum margin linear program using the matrix notation:

$$\begin{aligned} \underset{\hat{\mathbf{r}},\zeta}{\text{maximize}} \quad & \sum_i \zeta_i \\ \text{subject to} \quad & \zeta - (P^\pi - P^a) D^\pi \hat{\mathbf{r}} \leq 0 \\ & ||\mathbf{r}||_\infty \leq 1 \end{aligned} \tag{IRL-LP}$$

where $D^\pi = (I + \gamma(I - \gamma P^\pi)^{-1} P^\pi)$.

Average IRL. The standard maximum margin IRL problem infers a reward function of a single expert policy. However, it cannot learn from more than one policy. To apply it to our setting, we infer a rubeward function $\hat{r}_i$

for each of the demonstrations π_i^*. Then for any new reward function r_{eval}, we add the average of our inferred rewards, i.e., we optimize for $r_{\text{eval}}(s)+\frac{1}{k}\sum_i \hat{r}_i(s)$. So, implicitly, we assume the reward component of the inferred rewards is zero in expectation, and we can use the average to extract the constraint component of the inferred rewards.

Shared Reward IRL. We can also extend the maximum margin IRL problem to learn from multiple expert policies. For multiple policies, we can extend the maximum margin IRL problem to infer multiple reward function simultaneously:

$$\begin{aligned}
\underset{\hat{r}_1,\dots\hat{r}_n,\zeta}{\text{maximize}} \quad & \sum_i \zeta_i \\
\text{subject to} \quad & \hat{V}_1^{\pi_1}(s) - \hat{Q}_1^{\pi_1}(s,a) \geq \zeta_1, \quad \forall (s,a) \in S \times A \\
& \hat{V}_2^{\pi_2}(s) - \hat{Q}_2^{\pi_2}(s,a) \geq \zeta_2, \quad \forall (s,a) \in S \times A \\
& \vdots \\
& \hat{V}_k^{\pi_k}(s) - \hat{Q}_k^{\pi_k}(s,a) \geq \zeta_k, \quad \forall (s,a) \in S \times A
\end{aligned}$$

where we infer k separate reward functions $\hat{r}_i$ and $\hat{V}_i$ and $\hat{Q}_i$ are computed with reward function $\hat{r}_i$. For *Shared Reward IRL*, we model the reward functions as $\hat{r}_i(s) = \hat{r}_i^+(s) + \hat{c}(s)$, where $\hat{r}_i^+$ is different for each expert policy, and $\hat{c}$ is a shared constraint penalty. This gives us the following linear program to solve:

$$\begin{aligned}
\underset{\hat{\mathbf{r}}_1,\dots,\hat{\mathbf{r}}_k,\hat{\mathbf{c}},\zeta}{\text{maximize}} \quad & \sum_i \zeta_i \\
\text{subject to} \quad & \zeta_i - (P^{\pi_i} - P^a)D^{\pi_i}\hat{\mathbf{r}}_i - (P^{\pi_i} - \mathbf{P}^a)D^{\pi_i}\hat{\mathbf{c}} \leq 0 \text{ for each policy } i \\
& ||\hat{\mathbf{r}}_i||_\infty \leq 1 \text{ for each } i \\
& ||\hat{\mathbf{c}}||_\infty \leq 1
\end{aligned} \tag{IRL-LP-SR}$$

Known Reward IRL. Alternatively, we can assume we know the reward component, and only infer the constraint. In that case $\mathbf{r}_i$ is the known reawrd vector and no longer a decision variable, and we obtain the following LP:

$$\begin{aligned}
\underset{\hat{\mathbf{c}},\zeta}{\text{maximize}} \quad & \sum_i \zeta_i \\
\text{subject to} \quad & \zeta_i - (P^{\pi_i} - P^a)D^{\pi_i}\mathbf{c} \leq (P^{\pi_i} - P^a)D^{\pi_i}\mathbf{r}_i \text{ for each policy } i \\
& ||\hat{\mathbf{c}}||_\infty \leq 1
\end{aligned} \tag{IRL-LP-KR}$$

E.1.2 Maximum-Entropy IRL

We adapt Maximum Entropy IRL (Ziebart et al., 2008) to our setting by parameterizing the reward function as $\hat{r}(s,a) = \hat{\theta}_i^T \mathbf{f}(s,a) + \hat{\phi}^T \mathbf{f}(s,a)$. Here $\hat{\theta}_i$ parameterizes the "reward" component for each expert policy, and $\hat{\phi}$ parameterizes the reward function's shared "constraint" component.

Algorithm 2 shows how we modify the standard maximum entropy IRL algorithm with linear parameterization to learn $\hat{\theta}_i$ and $\hat{\phi}$ simultaneously. Instead of doing gradient updates on a single parameter, we do gradient updates for both $\hat{\theta}_i$ and $\hat{\phi}$ with two different learning rates α_θ and α_ϕ. Also, we iterate over all demonstrations $\pi_1^*, \dots, \pi_k^*$. This effectively results in updating $\hat{\phi}$ with the average gradient from all demonstrations, while $\hat{\theta}_i$ is updated only from the gradients for demonstration π_i^*.

Depending on initialization and learning rate, Algorithm 2 implements all of our IRL-based baselines:

- Average IRL: $\alpha_\theta > 0$, $\alpha_\phi = 0$, $\hat{\theta}_0 = 0$, $\hat{\phi}_0 = 0$
- Shared Reward IRL: $\alpha_\theta > 0$, $\alpha_\phi > 0$, $\hat{\theta}_0 = 0$, $\hat{\phi}_0 = 0$
- Known Reward IRL: $\alpha_\theta = 0$, $\alpha_\phi > 0$, $\hat{\theta}_i = \theta_i^{\text{known}}$, $\hat{\phi}_0 = 0$

For Known Reward IRL and Shared Reward IRL, we can apply the learned $\hat{\phi}$ to a new reward function by computing $r_{\text{eval}}(s,a) + \hat{\phi}^T \mathbf{f}(s,a)$. For Average IRL, we use $r_{\text{eval}}(s,a) + \frac{1}{k}\sum_i \hat{\theta}_i^T \mathbf{f}(s,a)$.

Algorithm 2 Maximum Entropy IRL, adapted to constraint learning.

Require: Expert policies $\pi_1^*, \dots, \pi_k^*$, learning rates α_θ, α_ϕ, initial parameters $\hat{\theta}_0, \hat{\phi}_0 \in \mathbb{R}^d$

Initialize: $\hat{\theta}_i \leftarrow \hat{\theta}_0$ for $i = 1, \dots, k$, $\hat{\phi} \leftarrow \hat{\phi}_0$, $i \leftarrow 0$
while not converged **do**
  $i \leftarrow \text{mod}(i + 1, k + 1)$
  Update policy $\hat{\pi}$ for $\hat{r}(s, a) = \hat{\theta}_i^T \mathbf{f}(s, a) + \hat{\phi}^T \mathbf{f}(s, a)$
  Compute the gradient: $\nabla = \mathbf{f}(\pi_i^*) - \mathbf{f}(\hat{\pi})$
  Update weights: $\hat{\theta}_i \leftarrow \hat{\theta}_i + \alpha_\theta \nabla$, $\hat{\phi} \leftarrow \hat{\phi} + \alpha_\phi \nabla$
return $\hat{\theta}_1, \dots, \hat{\theta}_k, \hat{\phi}$

Algorithm 3 Imitation learning baseline.

Require: Expert policies $\pi_1^*, \dots, \pi_k^*$, evaluation reward r_{eval}

$\hat{\mathcal{D}} \leftarrow \{\}$
for $i = 1, \dots, k$ **do**
  $\hat{\pi}_i^* \leftarrow$ `imitation_learning`(π_i^*)
  $\hat{\mathcal{D}} \leftarrow \hat{\mathcal{D}} \cup \{\hat{\pi}_i^*\}$
return $\text{argmax}_{\hat{\pi}_i^* \in \hat{\mathcal{D}}} G_{\text{eval}}(\hat{\pi}_i^*)$

E.2 Imitation Learning Baseline

Algorithm 3 shows the general imitation learning baseline discussed in the main paper. We can implement it using any imitation learning algorithm. The approach simply runs imitation learning on each demonstration and returns the best policy for a new evaluation reward.

In our experiments, we use synthetic demonstrations. Hence, for each demonstration, we know exactly which policy produced it, which allows us to implement the idealized IL baseline presented in the main paper.

This idealized version of Algorithm 3 is guaranteed to return safe policies because all demonstrations are safe. Also, if we assume linear reward functions and have no task- or environment-transfer, this algorithm returns the same solutions as `CoCoRL`. For this restricted setting, we could also obtain convergence guarantees for this algorithm.

However, for task and environment transfer this is no longer true. Our experiments show that `CoCoRL` can, e.g., learn to drive straight from a set of demonstrations that always turn left or right. The IL baseline clearly fails this task, because it only imitates turning left or right. Further, our experiments show that `CoCoRL` can learn to generalize from defensive drivers to aggressive drivers. Again, the IL baseline fails because it directly learns policies that do not generalize.

F Experiment Details

In this section, we provide more details about our experimental setup in the Gridworld environments (Appendix F.1) and the driving environment (Appendix F.2). In particular, we provide details on the environments, how we solve them, and the computational resources used.

F.1 Gridworld Experiments

Environment Details. An $N \times N$ Gridworld has N^2 discrete states and a discrete action space with 5 actions: *left*, *right*, *up*, *down*, and *stay*. Each action corresponds to a movement of the agent in the grid. Given an action, the agent moves in the intended direction except for two cases: (1) if the agent would leave the grid, it instead stays in its current cell, and (2) with probability p, the agent takes a random action instead of the intended one.

In the Gridworld environments, we use the state-action occupancies as features and define rewards and costs independently per state. We uniformly sample n_{goal} and n_{limited} goal tiles and limited tiles, respectively. Each goal cell has an average reward of 1; other tiles have an average reward of 0. Constraints are associated

with limited tiles, which the agent must avoid. We uniformly sample the threshold for each constraint. We ensure feasibility using rejection sampling, i.e., we resample the thresholds if there is no feasible policy. In our experiments, we use $N = 10$, $n_{\text{goal}} = 20$, $n_{\text{limited}} = 10$, and we have $n = 4$ different constraints. We choose discount factor $\gamma = 0.9$.

While the constraints are fixed and shared for one Gridworld instance, we sample different reward functions from a Gaussian with standard deviation 0.1 and mean 1 for goal tiles and 0 for non-goal tiles. To test reward transfer, we sample a different set of goal tiles from the grid for evaluation.

For the experiment without constraint transfer, we choose $p = 0.2$, i.e., a deterministic environment. To test transfer to a new environment, we use $p = 0.2$ during training and $p = 0$ during evaluation.

Linear Programming Solver. We can solve tabular environments using linear programming (Altman, 1999). The idea is to optimize over the occupancy measure $\mu_\pi(s, a)$. Then, the policy return is a linear objective, and CMDP constraints become additional linear constraints of the LP.

The other constraint we need is a Bellman equation on the state-action occupancy measure:

$$\begin{aligned}
\mu_\pi(s,a) &= \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty}\gamma^t \mathbb{1}_{\{s_t=s, a_t=a\}}\right] = \sum_{t=0}^{\infty}\gamma^t P(s_t = s, a_t = a|\pi) \\
&= \mu_0(s,a) + \gamma\sum_{t=0}^{\infty}\gamma^t P(s_{t+1} = s, a_{t+1} = a|\pi) \\
&= \mu_0(s,a) + \gamma\sum_{t=0}^{\infty}\gamma^t \sum_{s',a'} P(s|s',a')\pi(a|s)P(s_t = s', a_t = a'|\pi) \\
&= \mu_0(s,a) + \gamma\sum_{s',a'} P(s|s',a')\pi(a|s)\mu_\pi(s',a')
\end{aligned}$$

Using this, we can formulate solving a CMDP as solving the following LP:

$$\begin{aligned}
\underset{\mu_\pi(s,a)}{\text{maximize}} \quad & \sum_{s,a}\mu_\pi(s,a)r(s,a) \\
\text{subject to} \quad & \sum_{a}\mu_\pi(s,a) = \mu_0(s) + \gamma\sum_{s',a'}P(s|s',a')\mu_\pi(s',a'), \\
& \sum_{s,a}\mu_\pi(s,a)c_j(s,a) \le \xi_j, \text{ for each constraint } j \\
& \mu_\pi(s,a) \ge 0,
\end{aligned} \tag{CMDP-LP}$$

By solving this LP, we obtain the state-occupancy measure of an optimal policy μ_{π^*} which we can use to compute the policy as $\pi^*(a|s) = \mu_{\pi^*}(s,a)/(\sum_{a'}\mu_{\pi^*}(s,a'))$.

Computational Resources. We run each experiment on a single core of an Intel(R) Xeon(R) CPU E5-2699 v3 processor. The experiments generally finish in less than 1 minute, often finishing in a few seconds. For each of the 3 transfer settings, we run 100 random seeds for 26 different numbers of demonstrations k, i.e., for each method, we run 7800 experiments.

F.2 Driving Experiments

Environment Details. We use the intersection environment provided by `highway-env` (Leurent, 2018) shown in Figure 6. The agent controls the green car, while the environment controls the blue cars. The action space has high-level actions for speeding up and slowing down and three actions for choosing one of three trajectories: turning left, turning right, and going straight. The observation space for the policy contains the position and velocity of the agent and other cars.

The default environment has a well-tuned reward function with positive terms for reaching a goal and high velocity and negative terms to avoid crashes and staying on the road. We change the environment to have a reward function that rewards reaching the goal and includes driver preferences on velocity and heading angle

and multiple cost functions that limit bad events, such as driving too fast or off the road. To define the reward and cost functions, we define a set of features:

$$\mathbf{f}(s,a) = \begin{pmatrix} f_{\text{left}}(s,a) \\ f_{\text{straight}}(s,a) \\ f_{\text{right}}(s,a) \\ f_{\text{vel}}(s,a) \\ f_{\text{heading}}(s,a) \\ f_{\text{toofast}}(s,a) \\ f_{\text{tooclose}}(s,a) \\ f_{\text{collision}}(s,a) \\ f_{\text{offroad}}(s,a) \end{pmatrix} \in \mathbb{R}^9$$

where $f_{\text{left}}, f_{\text{straight}}, f_{\text{right}}$ are 1 if the agent reached the goal after turning left, straight, or right respectively, and 0 else. f_{vel} is the agents velocity and f_{heading} is its heading angle.

The last four features are indicators of undesired events. f_{toofast} is 1 if the agent exceeds the speed limit, f_{tooclose} is 1 if the agent is too close to another car, $f_{\text{collision}}$ is 1 if the agent has collided with another car, and f_{offroad} is 1 if the agent is not on the road.

The ground truth constraint in all of our experiments is defined by:

$$\begin{aligned} \phi_1 &= (0,0,0,0,0,1,0,0,0)^T \\ \phi_2 &= (0,0,0,0,0,0,1,0,0)^T \\ \phi_3 &= (0,0,0,0,0,0,0,1,0)^T \\ \phi_4 &= (0,0,0,0,0,0,0,0,1)^T \end{aligned}$$

and thresholds

$$\xi_1 = 0.2, \quad \xi_2 = 0.2, \quad \xi_3 = 0.05, \quad \xi_4 = 0.1,$$

i.e., we have one constraint for each of the last four features, and each constraint restricts how often this feature can occur. For example, collisions should occur in fewer than 5% of steps and speed limit violations in fewer than 20% of steps.

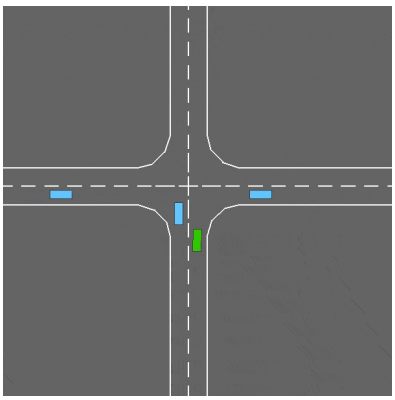


Figure 6: `highway-env` intersection environment.

The driver preferences are a function of the first five features. Each driver wants to reach one of the three goals (sampled uniformly) and has a preference about velocity and heading angle sampled from $\theta_{\text{vel}} \sim \mathcal{N}(0.1, 0.1)$, and $\theta_{\text{heading}} \sim \mathcal{N}(-0.2, 0.1)$. These preferences simulate variety between the demonstrations from different drivers.

When inferring constraints, for simplicity, we restrict the feature space to $(f_{\text{toofast}}, f_{\text{tooclose}}, f_{\text{collision}}, f_{\text{offroad}})$. Our approach also works for all features but needs more (and more diverse) samples to learn that the other features are safe.

To test constraint transfer to a new reward function, we sample goals in demonstrations to be either f_{left} or f_{right}, while during evaluation the goal is always f_{straight}.

To test constraint transfer to a modified environment, we collect demonstrations with more defensive drivers (that keep larger distances to other vehicles) and evaluate the learned constraints with more aggressive drivers (that keep smaller distances to other vehicles).

Driving Controllers. We use a family of parameterized driving controllers for two purposes: (1) to optimize over driving policies, and (2) to control other vehicles in the environment. The controllers greedily decide which trajectory to choose, i.e., choose the goal with the highest reward, and they control acceleration via a linear equation with parameter vector $\omega \in \mathbb{R}^5$. As the vehicles generally follow a fixed trajectory, these controllers behave similarly to an *intelligent driver model* (IDM). Specifically, we use linearized controllers proposed by in Leurent et al. (2018); see their Appendix B for details about the parameterization.

Constrained Cross-Entropy Method Solver. We solve the driving environment by optimizing over the parametric controllers, using a constrained cross-entropy method (CEM; Rubinstein and Kroese, 2004; Wen and

Algorithm 4 Cross-entropy method for (constrained) RL, based on Wen and Topcu (2018).

Require: n_{iter}, n_{samp}, n_{elite}
Initialize policy parameters $\mu \in \mathbb{R}^d$, $\sigma \in \mathbb{R}^d$.
for iteration $= 1, 2, \dots, n_{\text{iter}}$ **do**
  Sample n_{samp} samples of $\omega_i \sim \mathcal{N}(\mu, \text{diag}(\sigma))$
  Evaluate policies $\omega_1, \dots, \omega_{n_{\text{samp}}}$ in the environment
  if constrained problem **then**
    Compute number of constraints violated $N_{\text{viol}}(\omega_i) = \sum_j \mathbb{1}_{\{J_j(\omega_i)>0\}}$
    Compute total constraint violation $T_{\text{viol}}(\omega_i) = \sum_j \max(J_j(\omega_i), 0))$
    Sort ω_i in descending order first by N_{viol}, then by T_{viol}
    Let E be the first n_{elite} policies
    if $N_{\text{viol}}(\omega_{n_{\text{elite}}}) = 0$ **then**
      Sort $\{\omega_i | N_{\text{viol}}(\omega_i) = 0\}$ in descending order of return $G(\omega_i)$
      Let E be the first n_{elite} policies
  else
    Sort ω_i in descending order of return $G(\omega_i)$
    Let E be the first n_{elite} policies
  Fit Gaussian distribution with mean μ and diagonal covariance σ to E
return μ

Topcu, 2018). The constrained CEM ranks candidate solutions first by constraint violations and then ranks feasible solutions by reward. It aims first to find a feasible solution and then improve upon it within the feasible set. We extend the algorithm by Wen and Topcu (2018) to handle multiple constraints by ranking first by the number of violated constraints, then the magnitude of the violation, and only then by reward. Algorithm 4 shows the pseudocode for this algorithm.

Computational Resources. The computational cost of our experiments is dominated by optimizing policies using the CEM. The IRL baselines are significantly more expensive because they need to do so in the inner loop. To combat this, we parallelize evaluating candidate policies in the CEM, performing 50 roll-outs in parallel. We run experiments on AMD EPYC 64-Core processors. Any single experiment using `CoCoRL` finishes in approximately 5 hours, while any experiment using the IRL baselines finishes in approximately 20 hours. For each of the 3 constraint transfer setups, we run 5 random seeds for 30 different numbers of demonstrations k, i.e., we run 450 experiments in total for each method we test.

G Additional Results

In this section, we present additional experimental results that provide more insights on how `CoCoRL` scales with feature dimensionality and how it compares to all variants of the IRL baselines.

G.1 Validating `CoCoRL` in Single-state CMDPs

As an additional simple test-bed, we consider a single state CMDP, where $S = \{s\}$ and $A = \mathbb{R}^d$. We remove the complexity of the transition dynamics by having $P(s|s,a) = 1$ for all actions. Also, we choose $\gamma = 0$, and the feature vectors are simply $\mathbf{f}(s,a) = a$. Given a reward function parameterized by $\theta \in \mathbb{R}^d$ and constraints parameterized by $\phi_1, \dots, \phi_n \in \mathbb{R}^d$ and thresholds $\xi_1, \dots, \xi_n$, we can find the best action by solving the linear program $a^* \in \text{argmax}_{\phi_1^T a \leq \xi_1, \dots, \phi_n^T a \leq \xi_n} \theta^T a$.

To generate demonstrations, we sample both the rewards $\theta_1, \dots, \theta_k$ and the constraints $\phi_1, \dots, n$ from the unit sphere. We set all thresholds $\xi_j = 1$. The constraints are shared between all demonstrations.

Our results confirm that `CoCoRL` guarantees safety. We do not see any unsafe solution when optimizing over the inferred safe set. Figure 8 shows the reward we achieve as a function of the number of demonstrations for different dimensions d and different numbers of true constraints n. `CoCoRL` achieves good performance with a small number of samples and eventually approaches optimality. The number of samples depends on the dimensionality, as Theorem 3 suggests. We find the sample complexity depends less on the number of constraints in the true

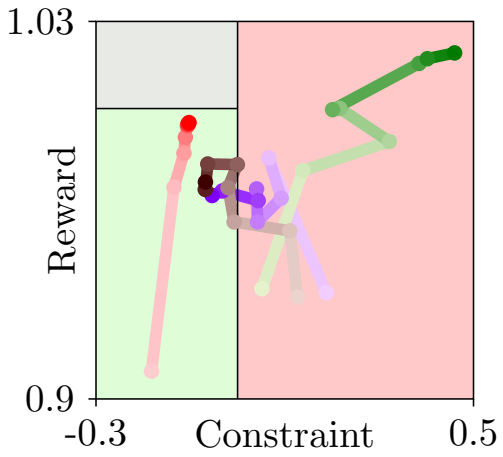


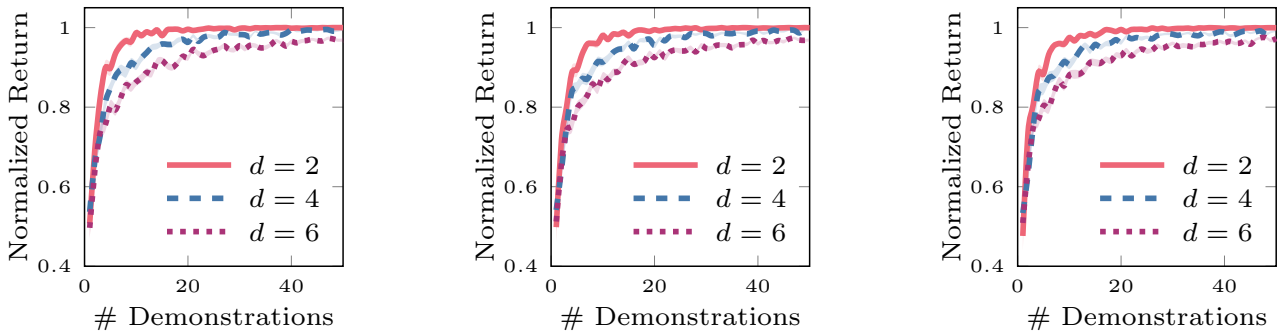
Figure 7: Illustration of safety and reward of CoCoRL, Average IRL, Shared Reward IRL, and Known Reward IRL in the Gridworld without constraint transfer. We show the 95th percentile of the constraint and the corresponding reward. The lines are colored from light to dark to indicate increasing number of demonstrations. The green region is safe, the red region is unsafe, and the grey region would be safe but is unattainable. CoCoRL quickly reaches a safe and near-optimal solution. Known Reward IRL achieves a high reward but is unsafe. Shared Reward IRL and Average IRL are first unsafe and then safe but suboptimal.

environment which is consistent with Theorem 3.

G.2 Maximum Margin IRL in Gridworld Environments

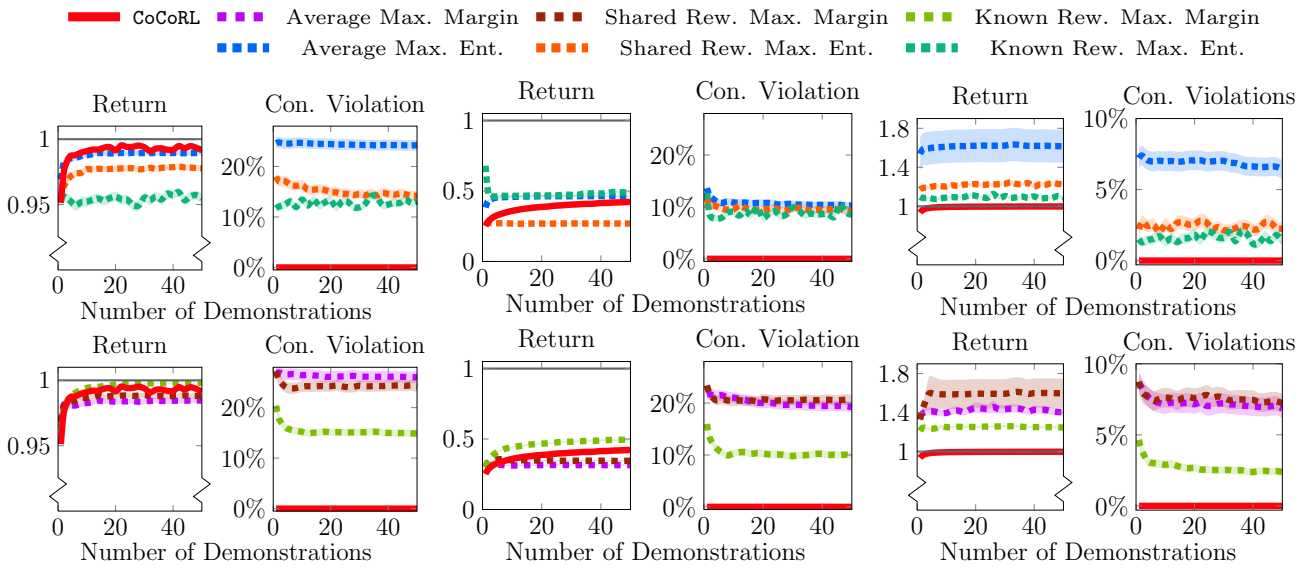
In addition to IRL baselines based on maximum entropy IRL, we evaluated methods based on maximum margin IRL (Appendix E.1.1). Figure 9 shows results from a 3×3 gridworld. The environment uses $N = 3$, $n_{\text{goal}} = 2$, $n_{\text{limited}} = 3$, and we have $n = 2$ different constraints.

Like the maximum entropy IRL methods, the maximum margin IRL methods fail to return safe solutions. Figure 7 illustrates how the different methods trade-off safety and maximizing rewards.



(a) $n = 8$ constraints (b) $n = 12$ constraints (c) $n = 16$ constraints

Figure 8: Return achieved by CoCoRL in single-state CMDPs for different dimensions d and numbers of constraints n. We plot the mean and standard errors over 100 random seeds, although the standard errors are nearly 0 everywhere. As expected, we need more demonstrations to approximate the true safe set well in higher dimensional feature spaces. There is no noticeable difference in sample complexity for different numbers of constraints. All solutions returned by CoCoRL are safe.



(a) Single environment. (b) Transfer to new task. (c) Transfer to new environment.

Figure 9: Experimental results in 3×3 Gridworld environments, with maximum entropy and maximum margin IRL baselines. The top row of plots shows CoCoRL compared to the maximum entropy IRL baselines. The bottom row of plots shows CoCoRL compared to the maximum-margin IRL baselines (Appendix E.1.1). The plots show mean and standard errors over 100 random seeds. The results are qualitatively similar: the IRL baselines are not safe and can return poor solutions. The maximum margin IRL baselines are slightly better than the maximum entropy IRL baselines. in terms of return but worse in terms of safety.