Learning Constraints from Offline Demonstrations via Superior Distribution Correction Estimation

Guorui Quan [1] **Zhiqiang Xu** [2] **Guiliang Liu** [1 *]

Abstract

An effective approach for learning both safety constraints and control policies is Inverse Constrained Reinforcement Learning (ICRL). Previous ICRL algorithms commonly employ an *online* learning framework that permits unlimited sampling from an interactive environment. This setting, however, is infeasible in many realistic applications where data collection is dangerous and expensive. To address this challenge, we propose Inverse Constrained Superior Distribution Correction Estimation (ICSDICE) as an offline ICRL solver. ICSDICE extracts feasible constraints from superior distributions, thereby highlighting policies with expert-exceeding rewards maximization ability. To estimate these distributions, ICSDICE solves a regularized dual optimization problem for safe control by exploiting the observed reward signals and expert preferences. Striving for transferable constraints and unbiased estimations, ICSDICE actively encourages sparsity and incorporates a discounting effect within the learned and observed distributions. Empirical studies show that ICSDICE outperforms other baselines by accurately recovering the constraints and adapting to high-dimensional environments. The code is available at `https://github.com/quangr/ICSDICE`.

1. Introduction

To deploy Reinforcement Learning (RL) algorithms for safety-critical applications, the control policy must adhere to underlying constraints in the environment (Liu et al., 2021). However, in many real-world tasks, due to the inherent complexity of environmental dynamics, the optimal constraint is often time-varying, context-sensitive, and rooted in human experience. These constraints are challenging to specify manually with prior knowledge and may not be readily available to RL agents in policy learning.

To address these problems, Inverse Constraint Reinforcement Learning (ICRL) considers inferring constraints from expert demonstrations. Existing ICRL algorithms (Scobee & Sastry, 2020; Malik et al., 2021; Liu & Zhu, 2022; Gaurav et al., 2023; Papadimitriou et al., 2023; Qiao et al., 2023; Xu & Liu, 2023) follow an online learning paradigm, where the agent explores and collects experience from an interactive environment. However, boundless exploration often involves unsafe behaviors that potentially violate underlying constraints. Such a shortcoming is fundamental since constraint violation contradicts the primary goal of safe control and may cause significant loss in practical applications, especially for algorithms that require a large number of training samples.

To effectively overcome these limitations, we propose designing an offline ICRL algorithm that relies solely on a fixed dataset for constraint inference. This task is challenging, primarily due to 1) the offline dataset covers only partial knowledge of the training environment, necessitating the algorithm to handle the uncertainty in unvisited states. 2) ICRL algorithms typically depend on executing policy rollout in an online manner to learn the constraint, making it infeasible to transfer to offline settings. 3) Offline ICRL is inherently ill-posed since the ground-truth constraint is not identifiable (Liu et al., 2022) from a limited dataset. How to specify the ICRL objective in a meaningful and practical way remains a critical challenge.

In this work, we address the aforementioned challenges by reformulating offline ICRL into a two-stage optimization problem, integrating policy learning with constraint inference through the estimation of superior distributions. These distributions capture high-risk behaviors that yield greater rewards than those achieved by expert agents. To estimate these distributions from the offline dataset, in the *forward* stage, we design a dual optimization problem for offline RL under a Constrained Markov Decision Process (CMDP). The problem solver constitutes an effective and meaningful

[*]Corresponding Author [1]School of Data Science, The Chinese University of Hong Kong, Shenzhen, Guangdong, 518172, P.R. China [2]Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. Correspondence to: Guiliang Liu <liuguiliang@cuhk.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

representation of the superior distributions. Subsequently, in the *inverse* stage, we update the constraint by assigning costs to the high-risk region sampled from these estimated distributions. We alternatively perform the forward and backward optimizations until superior distributions match expert demonstrations.

In the practical implementation, we propose an Inverse Constrained Superior Distribution Correction Estimation (ICSDICE) algorithm. ICSDICE actively encourages sparsity in the estimation of distributions, thereby capturing the minimum constraint necessary to account for expert preferences. This technique accommodates the ill-posed natures of ICRL problems and substantially improves the generalizability of the learned constraints. To stabilize training, ICSDICE utilizes the discounted probabilities to represent visitation distributions, which provides an unbiased estimation of Lagrangian multiplier.

Through comparative analysis with baseline methods in both discrete and continuous settings, we find ICSDICE can consistently learn a safe policy in conjunction with a sparse cost function. To better illustrate the advantages of ICSDICE, we conduct ablation studies to evaluate its performance under different configurations and evaluate how well the constraints can transfer to new environments.

2. Related Work

We review prior works that are most related to our approach.

Inverse Constrained Reinforcement Learning (ICRL). Prior ICRL methods extend the maximum entropy framework (Ziebart et al., 2008) for learned constraints from both the expert demonstrations and the interactive MDP environment. In the discrete state-action space, some recent research (Scobee & Sastry, 2020; McPherson et al., 2021) inferred the constrained sets for recording infeasible state-action pairs in Constrained MDP, but these studies were restricted to the environments with known dynamics. A subsequent work (Malik et al., 2021) extended this approach to continuous state-action spaces with unknown transition models by utilizing neural networks to approximate constraints. To enable constraint inference in stochastic environments, (Papadimitriou et al., 2023) inferred probability distributions over constraints by utilizing the Bayesian framework, and (Baert et al., 2023) incorporate the maximum causal entropy objective (Ziebart et al., 2010) into ICRL. Some recent works explore ICRL under different settings, e.g., (Gaurav et al., 2023) extended ICRL to infer soft constraints. Other works explored ICRL under the multi-agent (Liu & Zhu, 2022; 2023) and the meta learning (Liu & Zhu, 2024) setting. Striving for efficient comparisons, (Liu et al., 2023) established an ICRL benchmark across various RL domains. However, these algorithms primarily target online ICRL that utilizes interactive environments instead of offline datasets.

Offline Reinforcement Learning. Offline RL utilizes a data-driven RL paradigm where the agent learns the control policy exclusively from static datasets of previously collected experiences (Levine et al., 2020). To mitigate the distributional shift between training samples and testing data, previous offline RL solutions commonly involve constraining the learned policy to the data-collecting policy (Fujimoto et al., 2019; Kumar et al., 2019), making conservative estimates of future rewards (Kumar et al., 2020; Yu et al., 2021), and developing uncertainty-aware action selector (Janner et al., 2019; Kidambi et al., 2020). Some recent advancements on Offline RL (Sikchi et al., 2024; Xu et al., 2023) studied a regularized policy optimization problem with convex objective and linear constraints. Some recent DICE-related algorithms (Nachum et al., 2019a;b; Lee et al., 2021) solve offline RL by estimating the visitation distribution corrections term of the optimal policy. A recent IRL work (Yue et al., 2023) considered recovering conservative rewards from offline datasets, but none of these methods has studied the offline constraint inference.

3. Problem Formulation

Constrained Reinforcement Learning (CRL). A CRL problem is commonly based on a stationary Constrained Markov Decision Process (CMDP) $\mathcal{M} \cup \mathrm{c} := (\mathcal{S}, \mathcal{A}, p_{\mathcal{T}}, r, \mathrm{c}, \epsilon, \rho_0, \gamma)$ where: 1) $\mathcal{S}$ and $\mathcal{A}$ denote the space of states and actions. 2) $p_{\mathcal{T}} \in \Delta^{\mathcal{S}}_{\mathcal{S}\times\mathcal{A}}$[1] defines the transition distributions. 3) $r : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ and $\mathrm{c} : \mathcal{S}\times\mathcal{A} \to [0, C_{\max}]$ denotes the reward and cost functions. 4) $\epsilon \in \mathbb{R}_+$ denotes the bound of cumulative costs. 5) $\rho_0 \in \Delta^{\mathcal{S}}$ denotes the initial states distribution. 6) $\gamma \in [0, 1)$ is the discount factor. The goal of CRL policy $\pi \in \Delta^{\mathcal{A}}_{\mathcal{S}}$ is to maximize the expected discounted rewards under known constraints

$$\arg\max_{\pi} \mathbb{E}_{p_{\mathcal{T}},\pi,\rho_0}\Big[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\Big] \tag{1}$$

$$\text{s.t. } \mathbb{E}_{p_{\mathcal{T}},\pi,\rho_0}\Big[\sum_{t=0}^{T} \gamma^t c(s_t, a_t)\Big] \le \epsilon \tag{2}$$

Following the setting in (Malik et al., 2021), we are mainly interested in the hard constraints such that $\epsilon = 0$. Striving for clarity, we define the CMDP with the known cost as $\mathcal{M} \cup c$, and the CMDP without cost (i.e., CMDP$\backslash c$) as $\mathcal{M}$. Accordingly, the visitation distribution $d^\pi \in \Delta^{\mathcal{S}\times\mathcal{A}}$ (i.e., the normalized occupancy measure) produced by policy π can be denoted as:

$$d^\pi(s, a) = (1-\gamma)\pi(a|s)\sum_{t=0}^{\infty}\gamma^t p(s_t = s|\pi) \tag{3}$$

where $p(s_t = s|\pi)$ defines the probability of arriving state s at time step t by performing policy π.

[1] $\Delta^{\mathcal{X}}$ denotes the probabilistic simplex in the space $\mathcal{X}$, and $\Delta^{\mathcal{X}}_{\mathcal{Y}}$ denotes a function maps $\mathcal{Y}$ to $\Delta^{\mathcal{X}}$.

Inverse Constrained Reinforcement Learning. Note that traditional CRL problems often assume the constraint signals $c(\cdot)$ are directly observable from the environment, but in real-world problems, instead of observing the constraint signals, we often have access to expert demonstrations $\mathcal{D}_E$ that adhere to these constraints, and the agent is required to recover the constraint models from the dataset. This task is challenging because various combinations of rewards and constraints can explain the same expert demonstrations. Striving for the identifiability of solutions, ICRL algorithms (Malik et al., 2021; Liu et al., 2023; Papadimitriou et al., 2023) typically assume that *reward signals are observable and the goal is to recover only the constraints*, in contrast to Inverse Reinforcement Learning (IRL) (Ziebart et al., 2008), which aims to learn rewards from an unconstrained MDP.

From Online to Offline ICRL. Classic ICRL algorithm typically follows an online learning paradigm where the agent iteratively collects experience by interacting with the environment and using that experience for updating constraints and policy. Nevertheless, in many realistic settings, online interaction is impractical, either because data collection is expensive (e.g., in robotics, educational agents, or healthcare) or dangerous (e.g., in autonomous driving, or healthcare). To extend ICRL to the offline setting, we formally define the problem of *offline ICRL* as follows:

Definition 3.1. (Offline ICRL) In Offline ICRL, we are given an offline dataset $\mathcal{D}^O = \{s_n^O, a_n^O, r_n^O\}_{n=1}^{N_O}$. Within this dataset, expert trajectories are denoted as $\mathcal{D}^E = \{s_n^E, a_n^E, r_n^E\}_{n=1}^{N_E} \subset \mathcal{D}^O$. These expert trajectories are generated by an optimal policy π^E adhering to the unobserved constraints function $c(s,a)$. i.e. $\pi^E = \pi_c^* := \arg\max_\pi \mathbb{E}_{d^{\pi^E}}[r(s,a)]$, s.t. $\mathbb{E}_{d^{\pi^E}}[c(s,a)] \le \epsilon$. An estimated cost function $\hat{c}$ is a feasible solution to ICRL if and only if the optimal solution $\pi_{\hat{c}}^*$ can reproduce d^E. It is important to note that we cannot directly use the divergence between the expert distribution and the learned distribution as a feasible constraint for two reasons: 1. The divergence cannot be written as an inner product of a fixed cost function and the occupancy distribution d. 2. Such a constraint is not incorporated with reward information and thus cannot distinguish between high and low reward states.

In this work, we mainly study hard constraints with $\epsilon = 0$. This implies that the feasible set is bounded by the condition $d(s,a)c(s,a) \le 0$ for all state-action pairs. Accordingly, the CRL problem (2) can be equivalently formulated as:

$$\max_\pi \mathbb{E}_{d^\pi}[r(s,a)] \text{ s.t. } d^\pi(s,a)c(s,a) \le 0 \ \forall s,a \quad (4)$$

4. Inverse Constraint Inference via Superior Distribution Correction Estimation

In this section, we introduce our approach to 1) modeling the superior distributions (Section 4.1), 2) solving dual optimization problem (Section 4.2) to learn the distributions from the offline dataset and 3) inferring constraint (Section 4.3) based on these distributions.

4.1. Superior Distributions

To facilitate the cost identification, we formally define the superior distributions.

Definition 4.1. The set of superior distributions, denoted as $\mathcal{O}$, is defined as those distributions that generated by certain π and achieve higher cumulative rewards than expert, denoted as $\mathcal{O} = \{d^\pi : \mathbb{E}_{d^\pi}[r(s,a)] > \mathbb{E}_{d^E}[r(s,a)] \mid d^\pi(s,a) = \pi(a|s)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi)\}$.

We now establish the connection between superior distributions $\mathcal{O}$ and feasible solutions.

Proposition 4.2. *The $c(s,a)$ is a feasible solution for the ICRL problem if and only if 1) For every $d^* \in \mathcal{O}$, there exists at least one state-action pair (s,a) such that $c(s,a) > 0$ and $d^*(s,a) > 0$ and 2) if $d^E(s,a) > 0$, then $c(s,a) = 0$.*

Appendix C.3 shows the proof. Intuitively, for any superior distribution $d^* \in \mathcal{O}$, there's at least one state-action pair (s,a) with $d^*(s,a) > 0$ that constitutes a violation of the constraints. A direct approach to determine a feasible cost function c involves initially estimating the superior distributions set $\mathcal{O}$, followed by assigning a positive cost to at least one state-action pair in the support of each distribution d^* encompassed by this set.

However, directly constructing the set $\mathcal{O}$ is impractical, primarily due to 1) $\mathcal{O}$ may contain an infinite number of distributions, even in discrete state-action space. 2) identifying such sets necessitates solving an offline policy evaluation problem. Learning a low variance estimation of policy values is especially challenging in an offline context.

To circumvent these issues, we explicitly model a cost function $c(s,a)$ and iteratively update this function by aligning it with the expert demonstration. Specifically, we initialize cost function $c_0(s,a) = 0$. At each iteration k, based on $c_{k-1}(s,a)$, we compute a distribution d_k^* by solving the CRL problem (4). Since it guarantees that $d^* \in \mathcal{O}$ with the following proposition, we can maintain a subset of superior distributions by collecting the occupancy distribution of solved policies in each round: $\mathcal{O}_k = \{d_1^*, \dots, d_k^*\}$.

Proposition 4.3. *If the learned cost function $c_k(s,a)$ satisfied the condition that $d^E(s,a) > 0 \implies c_k(s,a) = 0$, the distribution generated by the CRL problem (4) d^* is in superior distributions, i.e., $d^* \in \mathcal{O}$.*

Appendix C.4 shows the proof. In the next step, we update $c_k(s,a)$ to ensure it's a valid cost function based on the distributions from previous k iterations $\mathcal{O}_k$. To achieve this, we increase cost in the support of these sampled distributions so that the next sampled distribution be closer to the expert distribution. This process terminates when d_k^* is sufficiently closed to d^E. Notably, our algorithm requires only solving a forward CRL problem (4) and a cost learning problem. It replaces the challenge of constructing the computationally-intractable set $\mathcal{O}$ with the task of matching distributions, which can be efficiently solved with a lower variance. In the ideal case, such a process can converge to the expert policy and will converge in a finite of steps in the discrete finite MDPs, as shown in Proposition 4.4.

Proposition 4.4. *If the MDP is finite, this process can recover d^E in a finite number of steps when problem 4 can be accurately solved.*

It's worth noting that a similar convergence analysis is shown in (Gaurav et al., 2023). However, they don't explicitly introduce the concept of superior distributions and rely on additional assumptions about the MDP.

The concept of superior distributions is key to inverse constraint reinforcement, we introduce the connection of our approach to adversarial imitation learning through the lens of superior distributions in Appendix B.

4.2. Offline Identification for Superior Distribution via Dual Optimization

In this section, we introduce the approach to identifying the superior distributions d^* under the offline learning setting. To achieve this goal, we regularize the cumulative rewards in the objective (4) with the offline demonstration. The new objective is given by:

$$J(d) = \mathbb{E}_{d(s,a)}[r(s,a)] - \xi_r D_f(d(s,a) \,||\, d^O(s,a)) \tag{5}$$

where d^O represents the visitation distribution in the offline dataset $\mathcal{D}^O$, and $D_f(\cdot \,||\, \cdot)$ denotes the f-divergence between two distributions. Intuitively, instead of maximizing only the reward, we augment the cumulative reward with a divergence regularizer to prevent it from deviating beyond the coverage of the offline data. Such an objective guarantees that the agent adheres to a conservative policy.

Since our goal is to identify the occupancy distribution, instead of estimating $\hat{\pi}$, we incorporate the Bellman flow constraint to the objective and directly solve the occupancy distribution d through a convex problem:

$$\max_{d(s,a)c(s,a)\le 0, d(s,a)\ge 0} \mathbb{E}_d[r(s,a)] - \xi_r D_f(d \,||\, d^O) \tag{6}$$
$$\text{s.t. } \textstyle\sum_{a\in\mathcal{A}} d(s,a) = (1-\gamma)\rho_0(s) + \gamma \sum_{s'\in\mathcal{S}, a'\in\mathcal{A}} d(s',a')p(s|s',a'),\ \forall s \in \mathcal{S}$$

To derive a solution for the problem (6), we introduce a Lagrangian dual variable V and construct a dual problem by following (Lee et al., 2021; Sikchi et al., 2024). Appendix C.1 shows the detailed derivation.

$$\min_V \max_d \mathbb{E}_d[\delta_V] - \xi_r D_f(d \,||\, d^O) + (1-\gamma)\mathbb{E}_{\rho_0}[V] \tag{7}$$
$$\text{s.t. } d(s,a)c(s,a) \le 0 \text{ and } d(s,a) \ge 0 \tag{8}$$

where

$$\delta_V(s,a) = r(s,a) + \sum_{s'\in\mathcal{S}} p_{\mathcal{T}}(s'|s,a)\gamma V(s') - V(s)$$

We provide the closed-form solution for the inner optimization problem in Equation (7) in the following :

Proposition 4.5. *Assume the learned distribution $d(s,a) > 0$ for all (s,a) such that $d^O(s,a) > 0$. The optimal solution for the inner optimization problem in (7), denoted as d^*, is given by :*

$$d^*(s,a) = d^O(s,a)\mathbb{1}_{c(s,a)=0} w_f^*(\delta_V(s,a)) \tag{9}$$

Substituting d^* into problem (7), the problem becomes:

$$\min_V \ \mathbb{E}_{d^O}\left[\xi_r \mathbb{1}_{c(s,a)=0} f_p^*\left(\frac{\delta_V(s,a)}{\xi_r}\right)\right] + (1-\gamma)\mathbb{E}_{\rho_0}[V] \tag{10}$$

where w_f^* and f_p^* is related to the convex function f specified in f-divergence: $w_f^*(y) = \max(0, f'^{-1}(\frac{y}{\xi_r}))$, $f_p^*(y) = \mathbb{E}_{w^*}[y] - f(w^*(y))$.

The proof is in Appendix C.2. In the representation of superior distributions d^* (Equation 9), ω^* denotes a correction term in Distribution Correction Estimation (DICE) (Nachum et al., 2019a), and its implementation is vital for controlling the sparsity of the estimated constraint (Section 5.1). At each interaction k, we estimate a d_k^*, and record it to the set $\mathcal{O}_k = \mathcal{O}_{k-1} \cup d_k^*$, with which we learn the cost function as follows.

4.3. Constraint Inference from Distribution Set

Given the set of estimated superior distributions $\mathcal{O}_k$, we assign high costs to state-action pairs in the support of these distributions (i.e., $\{(s,a) \mid d^*(s,a) > 0 \wedge d^* \in \mathcal{O}_k\}$) by following Proposition 4.2. To achieve this goal, we design the following objective to update the cost function.

$$c_k = \arg\min_c \mathbb{E}_{d^*\in\mathcal{O}_k}\left[\mathbb{E}_{d^*(s,a)}\left[(c(s,a)-1)^2)\right] + \psi(c)\right]$$
$$\text{s.t. } \mathbb{E}_{d^E(s,a)}[c(s,a)] = 0 \tag{11}$$

where we optimize the cost by adjusting mean cost to 1 for state-action pairs with positive probability in $d^* \in \mathcal{O}$ and ψ denotes a cost regularizer.

In our experiment, we introduce the L2 norm of the cost function as the cost regularizer, and by utilizing the Lagrange variable λ_e, the final objective can be defined as:

$$c_k = \arg\min_c \mathbb{E}_{d^*\in\mathcal{O}_k}\left[\mathbb{E}_{d^*}\left[(c(s,a)-1)^2 + \lambda_{\text{L2}} c(s,a)^2\right]\right] + \lambda_e \mathbb{E}_{d^E(s,a)}\left[c(s,a)^2\right] \tag{12}$$

5. Practical Implementation

In this section, we present the practical implementation of ICSDICE (see Algorithm 1) by introducing the approach to: 1) Encouraging sparsity in the learned constraints (Section 5.1), 2) Utilizing the discounted visitation distribution to enable the recovery of right sparse superior visitation distributions (Section 5.2), and 3) Extracting policy from the learned distribution (Section 5.3).

5.1. Encouraging the Sparsity of Inferred Constraints

The Importance of Sparsity. Similar to IRL, the optimal constraint in ICRL is not uniquely identifiable. indicating that multiple constraints may equivalently explain the expert behaviors. To address this issue, ICRL algorithms aim to learn *the most sparse constraint* under which the imitation agent can reproduce the behaviors of the expert (Scobee & Sastry, 2020). Additionally, *sparsity is critical for guaranteeing the applicability of the learned constraints*. This is because the lack of sparsity can lead to learning an excessively restrictive constraint that blocks all the movement not covered by the expert dataset. While such a constraint may accurately reproduce the observed expert behaviors, it lacks practical utility for two reasons: 1) The offline dataset may not cover all potential actions an expert agent could execute (i.e., $\{a \mid \pi^E(a|s) > 0\}$), thereby resulting in the exclusion of many feasible actions due to overly restrictive constraints. and 2) it cannot be generalized to environments with even minor modifications to rewards or dynamics (see experiments in Section 6.3), which is a critical challenge in bridging the Sim-to-Real gap in practice.

From Cost Sparsity to Distributional Sparsity. In accordance with Proposition 4.2, positive costs are allocated exclusively to the state-action pairs within the set $\{(s,a) \mid d^*(s,a) > 0 \wedge d^* \in \mathcal{O}\}$. Consequently, when the estimated set superior distributions d^* are sparse, meaning there are relatively few state-action pairs where $d^*(s,a) > 0$, the positive costs will be assigned to a smaller number of state-action pairs, which efficiently encourages the sparsity of constraint.

To learn a more sparse distribution, we utilize the chi-square distance as the f-divergence metric D_f in objective (7). Specifically, the corresponding convex function f and the superior distributions are represented as:

$$f(x) = (x-1)^2 \text{ and } f'^{-1}(x) = 1 + \frac{x}{2} \tag{13}$$

$$\frac{d^*(s,a)}{d^O(s,a)} = w_f^*(s,a) = \left(1 + \frac{\delta_V(s,a)}{2\xi_r}\right)_+ \tag{14}$$

This representation can facilitate sparsity, primarily because in the cost learning step, if the temporal difference $\delta_V(s,a) \le -2\xi_r$, objective (14) ensures $d^*(s,a) = 0$ for this (s,a), indicating the following movements will not influence the costs. As a result, positive costs can *not* be assigned to the low-reward state-action pairs, which encourages the sparsity of $c(s,a)$. By reducing ξ_r (i.e., the tolerant threshold), the impact of this clipping mechanism becomes more influential. In this way, we can control the sparsity of the learned distribution by adjusting ξ_r. On the other hand, this method makes the cost function only sensitive to regions exhibiting constraint-violating behaviors, particularly those associated with rewards exceeding the expert level.

5.2. Utilizing the Discounted Visitation Distribution

While previous DICE-related algorithms (Kostrikov et al., 2020; Lee et al., 2021; Ma et al., 2022a) commonly sample data by following the *empirical visitation distribution* d^O in the offline dataset $\mathcal{D}^O$, we propose utilizing a *discounted* visitation distribution in our training objective (10).

Definition 5.1. Let π^O denote the sampling policy for the offline dataset $\mathcal{D}^O$. The discounted visitation distribution is represented as:

$$d_\gamma^O(s,a) = \sum_{k=0}^{\infty} \gamma^k P(s,a|\rho_0,\pi^O) \tag{15}$$

After replacing $d^O(s,a)$ with $d_\gamma^O(s,a)$ in our previous objective (10), we obtain a new objective for updating the value functions:

$$\mathbb{E}_{d_\gamma^O}\left[\xi_r \mathbb{1}_{c(s,a)\le 0} f_p^*(\frac{\delta_V(s,a)}{\xi_r})\right] + (1-\gamma)\mathbb{E}_{\rho_0(s)}[V(s)] \tag{16}$$
$$= \mathbb{E}_{d_\gamma^O}\left[\xi_r \mathbb{1}_{c(s,a)\le 0} f_p^*(\frac{\delta_V(s,a)}{\xi_r}) + (1-\gamma)\frac{\rho_0(s)}{d_\gamma^O(s)}V(s)\right]$$

Such a replacement offers several benefits: 1) The discount factor is crucial in defining the occupancy measure; thus, incorporating this factor into the construction of the visitation distribution is essential. 2) It introduces a more principled approach to managing values in absorbing states. To elucidate, we may interpret the left term in the preceding objective as follows:

$$\mathbb{E}_{d_\gamma^O}\left[\xi_r \mathbb{1}_{c(s,a)=0} f_p^*(\frac{\delta_V(s,a)}{\xi_r})\right] = \tag{17}$$
$$\sum_{t=0}^{T-1} \lambda^t \mathbb{1}_{c(s_t,a_t)=0} f_p^*\left(\frac{\gamma V(s_{t+1}) + r(s_t,a_t) - V(s^t)}{\alpha}\right)$$
$$+ \sum_{t=T}^{\infty} \lambda^t f_p^*\left(\frac{\gamma V(s_T) - V(s_T)}{\alpha}\right)$$

where s_T denotes an absorbing state occurring at time T. The value of s_T is updated to maximize $\frac{\lambda^T}{1-\lambda} f_p^*[\frac{(\gamma-1)}{\alpha}V(s_T)]$ as opposite to the previous methods. For example, striving for simplicity, OptiDICE (Lee et al., 2021) enforces $V(s_T)$ to zero. This conventional approach is inadequate because V serves as a Lagrangian multiplier (see objective 6) in the optimization problem.

To demonstrate the importance of our update, we conducted experiments in the Point Maze environment(Fu et al., 2020) to examine whether the algorithm can identify the superior distribution d^* under the offline learning setting. We demonstrated the effects of our modified learning objective against OptiDICE (Lee et al., 2021) and the ablation of our algorithm that utilizes the undiscounted $d^O(s,a)$. Figure 1 visualize the learned distribution. While our ICSDICE objective can learn a sparse distribution capturing the shortest path to the destination, both OptiDICE and the ablated ICSDICE objective fail to achieve this goal due to their biased estimate. Besides, we find as ξ_r decreases, the distribution learned by our ICSDICE objective becomes more sparse, which well aligns with our intuition in Section 5.1.

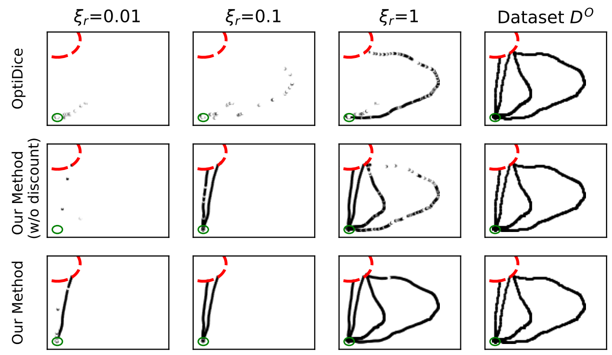

Figure 1: A 2D maze example for comparing different DICE objectives, varying different ξ_r. We use a green circle to mark the start point, and a red line represents the finish line.

5.3. Extracting Policy from the Learned Distribution

Once we learned a matched distribution d^* with d^E, the policy π defined by optimal distribution d^* is given by $\pi(a|s) = \frac{d(s,a)}{\sum_{a'} d(s,a')}$. We can extract the policy by maximizing $\mathbb{E}_{d^*(s,a)}[\log \pi(a|s)] = \mathbb{E}_{d^O(s,a)}[\omega^*(s,a) \log \pi(a|s)]$. To stabilize the training, we utilize the advantage-weighted regression (Peters & Schaal, 2007) method as follows:

$$\max \mathbb{E}_{d^O(s,a)} \left[e^{\delta_V(s,a)} \log \pi(a|s) \right] \quad (18)$$

We also use a max clipping operation on δ_V in case its values are too large, which could cause numeric instability during training.

6. Empirical Evaluation

6.1. Discrete Grid-world Environment

We utilize grid-world environments for evaluating our algorithm. These environments consist of 7x7 discrete maps, and each map has a unique constraint setting (First row in Figure 2). Within these environments, each agent is permitted to perform eight actions: moving up, down, right, left, or in one of four diagonal directions. The primary objective for every agent is to navigate from the start to the end point, taking the shortest possible route while avoiding specific constrained states. The agent receives a -1 reward for each step taken until it reaches its destination. To enable offline ICRL, we provide an expert dataset $\mathcal{D}^E$ and a sub-optimal dataset $\mathcal{D}^{\neg E}$ (collected through random walks) for each environment. The size of the sub-optimal dataset is 50 (trajectories) (Check details in Appendix A.1).

Algorithm 1 Inverse Constrained Superior Distribution Correction Estimation (ICSDICE)

```
Require: Offline dataset $\mathcal{D}^O = \{\mathcal{D}^E, \mathcal{D}^{\neg E}\}$, Running iterations $K$; Initialize $c_0(\cdot) = 0, \mathcal{O}_0 = \{\}$;
1: for $k = 1 \dots K$ iterations do
2:     Learn the value function $V_k$ with the dual optimization objective (16) based on cost function $c_{k-1}$;
3:     Solve the distribution $d_k^*$ with objective (9);
4:     Update the distribution set $\mathcal{O}_k = \mathcal{O}_{k-1} \cup \{d_k^*\}$;
5:     Update the cost function $c_k(s,a)$ with objective (12).
6:     Extract policy $\pi$ with objective (18);
7: end for
```

Constraint Visualization. Figure 2 illustrates the information captured by different methods from an offline dataset. We find ICSDICE learns sparse constraints that are most similar to the ground-truth ones. When it comes to comparison methods, *Offline IL* (i.e., Inverse Q-learning (Al-Hafez et al., 2023)) learns a Q function representing reward. We observe that the value function in each state learned by the imitation model fails to accurately capture the constraint, especially in settings 1 and 4. This highlights that the constraint cannot be captured by traditional IRL algorithms. The method, *Offline IL + rewards*, is an extension of Inverse Q learning, where we incorporate rewards into the Q-learning process by modeling the learned rewards as $\bar{r}(s,a) = r(s,a) + c(s,a)$, where $r(s,a)$ represents the known rewards. This enables the cost signals to be extracted from the learned Q function. For additional details, please refer to Appendix A.4. We observe that after incorporating rewards, the learned value function becomes overly conservative and thus can only represent part of cost information. This indicates that incorporating reward signals into IRL is insufficient to learn a well-defined cost function.

Additionally, Figure 3 visualizes the learned superior distribution d_k^* and the learned cost function c_k from the learning process. Our method successfully samples sparse d^* from $\mathcal{O}$ and effectively learns a sparse cost function by capturing sparse superior distributions.

6.2. High-Dimensional Continuous Environment

We extend MuJoCo environments (Todorov et al., 2012) for evaluating ICRL algorithms by incorporating predefined

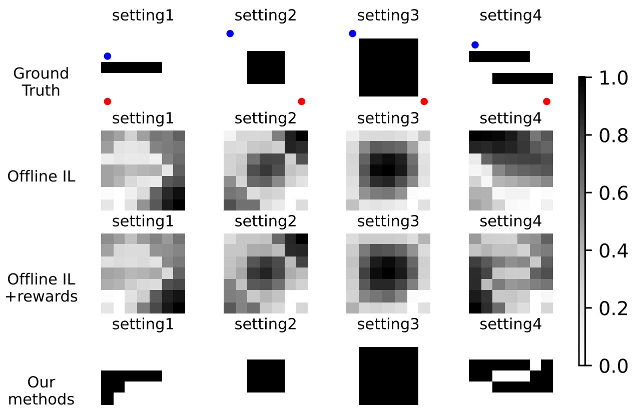

Figure 2: Visualizing the constraints captured by different methods based on an offline dataset. Blue, red mark the starting, target states.

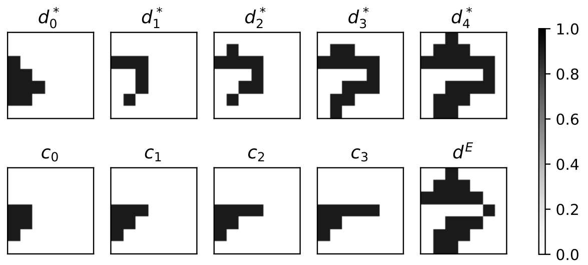

Figure 3: Visualizing the support of superior distributions (top) and constraints (bottom) in the setting 1 from the beginning (left) until the end (right) of training.

constraints into each environment. Inspired by (Liu et al., 2023), the *first* type of constraint, marked by **HalfCheetah(Obstacle)**, prohibits the robot from moving backward, although it's relatively easier for the robot to move backward than forward. The *second* type of constraint, denoted as **Ant(LS)** and **Walker(LS)**, limits the robot's maximum speed of moving forward. The *third* type of constraint, denoted as **Ant(Blocked)** and **Hopper(Blocked)**, restricts the robot leg's angular velocity and thus constricts the size of each move. For further details, please refer to Appendix A.2.

Experiment Setting. By following (Malik et al., 2021), we adopt the following evaluation metrics: 1) **Constraint Violation Rate**, which assesses the likelihood for a policy to violate a constraint within a given trajectory, and 2) **Feasible Cumulative Rewards**, which calculates the total rewards accumulated by the agent before violating any constraints. We run experiments with 5 different seeds and present the mean $\pm$ std results for each algorithm. Appendix A.5 reports the detailed settings and random seeds.

Comparison Methods. Due to the lack of offline ICRL baselines, we compare ICSDICE with the following baselines: 1) **LS-IQ** follows the implementation of *Least Square Inverse Q-Learning* (Al-Hafez et al., 2023) method that infers reward value functions from offline data to imitate expert policy. 2) **SMODICE** refers to the recently proposed *SMODICE* (Ma et al., 2022b) algorithm that leverages the dual and offline reward function to control. 3) **Behavior Cloning**. As these methods do not utilize reward information, we introduce two additional baselines: 1) **OptiDICE-Constraint** replaces our DICE objective in Section 5.2 with OptiDICE. 2) **SMODICE-Constraint** Based on SMODICE, we incorporate rewards information by adding environment rewards to the SMODICE's learned discriminator reward. For additional details, please refer to Appendix A.4.

Table 1 shows the evaluation results, and Figure 7 illustrates the corresponding learning curve is shown in. Across all environments, ICSDICE achieves leading performance in imitating high-performing policies offline while maintaining policy safety. The offline IL methods cannot ensure lower cost even when incorporated with rewards information. This limitation is understandable since these methods are not specifically designed for safety concerns.

One intriguing observation is that OptiDICE-Constraint performs comparably to our method. Given the tricks applied by OptiDICE-c (normalization and soft-chi divergence), it's not surprising that OptiDICE-c may perform slightly better than our algorithm in the task of recovering policy. However, it's important to note that this improvement comes at the cost of learning a dense superior distribution and the inaccurate constraint. To understand the difference to our ICSDICE , we investigate whether these ICRL methods have learned the true cost by utilizing the ROC (Receiver Operating Characteristic) curve to illustrate the relationship between the learned cost and the ground truth cost. As shown in Figure 5, although OptiDICE-Constraint exhibits similar performance in policy learning, our ICSDICE achieves *notably higher performance in cost identification*. This is because OptiDICE-Constraint tends to have a high false positive rate due to its biased estimation of the distribution (See Section 5.2). Additionally, as ξ_r decreases, the curve shifts to the left, indicating the learned constraint becomes more sparse and thus the false positive errors become smaller. For more evidence, Figure 8 visualizes the learned constraints.

6.3. Generalizability of Constraints

To examine whether the learned constraint can be transferred to new configurations, we apply our methods and OptiDICE-Constraint to learn cost functions in the limited-speed Ant environment. Then, we modify the environment setting by rewarding the velocity in a distinct direction. We utilize the forward objective (10) to learn policy based on these costs under new rewards. Ideally, the transfer behaviors should move in the rewarded direction while adhering to the same constraint on the x-axis velocity. For more details, please refer to Appendix A.3. Additionally, we used a zero-cost function (denoted as w/o cost) for comparison.
Figure 4 shows the learning curve of cumulative rewards

Table 1: MuJoCo testing performance. We report average cumulative rewards and costs over 10 runs. **Bold** denotes safe methods with a violation rate below 5%, gray denotes unsafe methods with a violation rate above 5%, and blue highlights methods whose standard deviation range includes the average performance of the top-performing method. The percentage in the dataset column indicates how many percent of expert transitions are known as expert demonstrations.

Dataset	Task	BC		LS-IQ		SMODICE		SMODICE-c		OptiDICE-c		Our Method	
		cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑
100% expert	Ant(Blocked)	**0.04±0.02**	**876±138**	0.40±0.12	-63±208	0.33±0.07	1,410±153	0.53±0.03	1,763±180	**0.01±0.00**	**3,070±91**	**0.01±0.00**	**3,073±103**
	Ant(LS)	**0.00±0.00**	**1,106±93**	0.07±0.06	-1,232±1,616	0.06±0.05	1,341±50	**0.03±0.04**	**1,361±67**	**0.00±0.00**	**1,368±3**	**0.00±0.00**	**1,340±44**
	HalfCheetah(Obstacle)	0.30±0.40	731±693	**0.04±0.05**	**2,175±775**	0.13±0.11	3,565±345	0.30±0.24	3,829±661	**0.03±0.06**	**2,749±597**	**0.04±0.04**	**2,315±740**
	Hopper(Blocked)	0.40±0.05	158±78	0.07±0.02	350±242	0.23±0.12	2,024±657	0.19±0.21	1,609±907	**0.00±0.00**	**1,255±576**	**0.02±0.02**	**1,017±341**
	Walker(LS)	**0.01±0.02**	**-7±5**	0.17±0.09	603±203	0.52±0.19	2,334±238	0.18±0.15	1,871±155	**0.01±0.01**	**1,538±283**	**0.01±0.02**	**1,587±308**
50% expert	Ant(Blocked)	**0.03±0.01**	**1,012±258**	0.35±0.16	-14±70	**0.04±0.03**	**2,856±266**	0.18±0.07	2,574±284	**0.01±0.00**	**3,069±141**	**0.01±0.00**	**3,025±101**
	Ant(LS)	**0.00±0.00**	**1,047±88**	0.10±0.00	-125±187	0.25±0.24	1,516±234	0.22±0.13	1,422±106	**0.00±0.00**	**1,361±9**	**0.00±0.00**	**1,364±5**
	HalfCheetah(Obstacle)	0.08±0.11	938±807	**0.03±0.04**	**2,097±365**	0.22±0.13	3,796±170	0.21±0.13	3,693±434	**0.01±0.02**	**2,086±1,153**	**0.00±0.01**	**1,657±723**
	Hopper(Blocked)	0.46±0.20	209±77	0.07±0.11	245±146	0.26±0.27	1,589±819	0.33±0.12	1,166±692	**0.04±0.03**	**1,327±885**	**0.02±0.01**	**1,101±714**
	Walker(LS)	**0.00±0.00**	**-9±3**	**0.02±0.03**	**123±146**	0.23±0.27	1,922±209	0.49±0.26	2,191±408	**0.01±0.03**	**1,645±73**	**0.05±0.07**	**1,168±516**

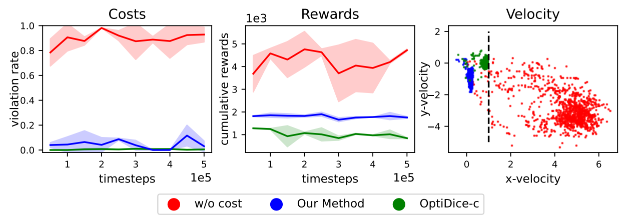

Figure 4: Generalization performance of learned costs. The black dashed line marks the velocity constraint.

and costs in the first two plots, and the velocity distribution sampled from the different transfer policies in the third plot. We observe that, with a zero-cost function, the policy can achieve high rewards but also has a high violation rate. The OptiDICE-Constraint method's learned cost resulted in the agent being constrained to only move forward and achieve the lowest return. This occurred because it masked unnecessary states, leading to an overly conservative cost and ultimately failing during transformation. In contrast, our methods successfully enabled the agent to follow the constraint while moving in the direction aligning with the rewards. These results demonstrate that our method can effectively learn a transferable cost function.

6.4. Realistic Environments

To evaluate the practical applicability of our algorithm, we examined its performance in various traffic scenarios represented by highway driving datasets HighD (Krajewski et al., 2018). Additionally, we utilized the commonroad-RL (Wang et al., 2021) environments to extract relevant features from driving scenarios.

We first conducted an experiment to determine if our algorithm could effectively identify and adhere to constraints related to vehicle velocity. We set the constraint as Car Velocity $\leq$ 40 m/s. Table 2 presents the performance of different algorithms. Our methods outperformed the baseline BC and OptiDICE-c in terms of both cumulative returns and Area Under the Curve (AUC).

To evaluate the performance on more challenging tasks involving high-dimensional inputs like images, we replaced the input to the constraint network with raw images and modified the network structure to use a CNN instead of an MLP. We designed a constraint related to car distance (Car distance $\geq$ 20m) and expected our method to learn this distance information from the pixels. We tested the learned cost function on a split test dataset, which included scenarios not seen during training. Our algorithm learned costs that could transfer to these settings, achieving an AUC of 0.75, while the OptiDICE-c baseline only achieved 0.69. Additionally, we visualized the images assigned with the highest cost in 5 scenarios in Figure 6. It can be clearly observed that our method assigns high costs to close car distances, while the OptiDICE-c baseline fails to do so.

Table 2: Performance under HighD speed constraint setting.

Metric	BC	OptiDice-c	Our Method	Expert
Return	-2.3 ± 2.3	-1.6 ± 3.7	$\mathbf{10.7 \pm 1.5}$	*14.0*
Cost	1.4%	12%	6.5%	*0.8%*
AUC	NA	0.76 ± 0.02	$\mathbf{0.81 \pm 0.01}$	NA

6.5. Ablation Study

In this experiment, we conduct ablations on our algorithm's design choice. We evaluate the performance of the learned policy by cumulative return and violation rate, and assess the learned cost discriminator by comparing the Area under the ROC curve. We compare the following variants: *w/o **discount***: Replace the discounted distribution with the empirical distribution. *w/o **constraint***: Remove the constraint learning step.

Table 3: Performance of different variants of ICSDICE.

Task	Our Method		w/o discount		w/o constraint	
	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑
Ant(Blocked)	**0.01**	**3,073**	**0.05**	**2,894**	0.21	1,692
Ant(LS)	**0.00**	**1,340**	**0.00**	**1,369**	0.18	1,381
HalfCheetah(Obstacle)	**0.04**	**2,315**	0.07	3,332	0.30	4,343
Hopper(Blocked)	**0.02**	**1,017**	**0.02**	**1,009**	0.32	2,091
Walker(LS)	**0.01**	**1,587**	0.13	1,256	0.29	1,942

The results in Table 3 and Table 4 indicate that, by using the

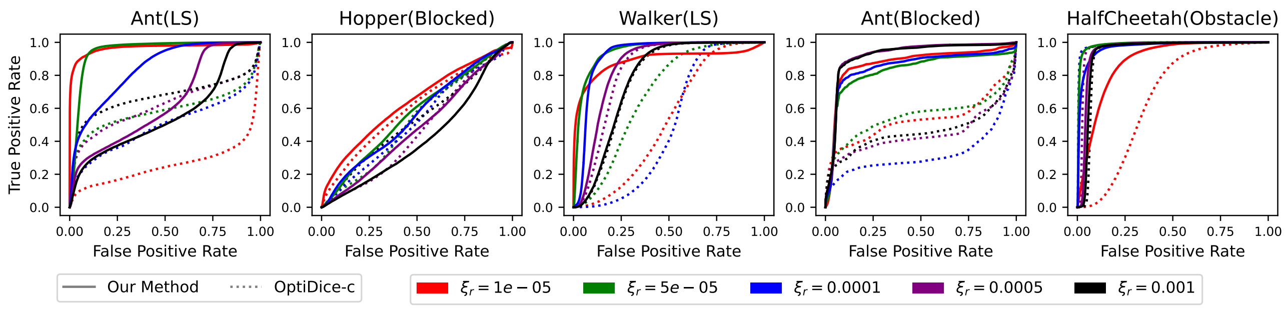

Figure 5: Receiver Operating Characteristic (ROC) Curve illustrating the accuracy of the learned cost. A curve closer to the top-left corner indicates higher accuracy.

Table 4: Area under the ROC Curve (AUC) of different variants of ICSDICE.

Task	Our Method	w/o discount	OptiDice-c
Ant(LS)	0.94±0.01	**0.98±0.01**	0.60±0.04
Ant(Blocked)	**0.91±0.01**	0.88±0.01	0.54±0.08
Hopper(Blocked)	**0.63±0.04**	**0.63±0.03**	0.59±0.01
HalfCheetah(Obstacle)	**0.98±0.02**	0.77±0.05	**0.97±0.02**
Walker(LS)	**0.93±0.02**	0.86±0.00	0.80±0.03

discounted distribution, our objectives can strike the right trade-off between the performance of the learned policy and the learned cost discriminator. Our approach performs very well on both the recovered policy and the discriminator performance compared to other ablation variants.

7. Conclusion

In this paper, we solve an offline ICRL by proposing a ICSDICE algorithm. As the first attempt, ICSDICE derives superior distributions from the dual objective of regularized policy learning under a CMDP and proposes the two-stage optimization method for updating both policies and constraints. To enhance practical applicability, ICSDICE effectively addresses insufficient sparsity and biased estimation. Empirical results demonstrate the performance of ICSDICE in various settings. A promising avenue for future research involves extending our method to accommodate uncertainty raised in the offline dataset and conducting a rigorous theoretical study for analyzing the convergence rate and sample efficiency of ICSDICE .

Acknowledgements

This work is supported in part by the National Key R&D Program of China under grant No.2022ZD0116004, Guangdong-Shenzhen Joint Research Fund under grant 2023A1515110617, GuangDong Basic and Applied Basic Research Foundation under grant 2024A1515012103, Shenzhen Fundamental Research Program (General Program) under grant JCYJ20230807114202005, Shenzhen Science and Technology Program ZDSYS20211021111415025, and Guangdong Provincial Key Laboratory of Mathematical Foundations for Artificial Intelligence (2023B1212010001).

Impact Statement

Our approach significantly advances the ability to acquire behavioral insights from datasets, furthering the progress of user-centric artificial intelligence systems. While there are numerous potential societal implications stemming from our research, we believe that none require specific emphasis in this context.

References

Al-Hafez, F., Tateo, D., Arenz, O., Zhao, G., and Peters, J. Ls-iq: Implicit reward regularization for inverse reinforcement learning. *arXiv preprint arXiv:2303.00599*, 2023.

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

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, volume 97, pp. 2052–2062, 2019.

Garg, D., Chakraborty, S., Cundy, C., Song, J., and Ermon, S. Iq-learn: Inverse soft-q learning for imitation. In *Neural Information Processing Systems (NeurIPS)*, pp. 4028–4039, 2021.

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

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Neural Information Processing Systems (NeurIPS)*, pp. 4565–4573, 2016.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Neural Information Processing Systems (NeurIPS)*, pp. 12498–12509, 2019.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Kostrikov, I., Nachum, O., and Tompson, J. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations (ICLR)*, 2020.

Krajewski, R., Bock, J., Kloeker, L., and Eckstein, L. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2118–2125, 2018. doi: 10.1109/ITSC.2018.8569552.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Neural Information Processing Systems (NeurIPS)*, pp. 11761–11771, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Lee, J., Jeon, W., Lee, B., Pineau, J., and Kim, K.-E. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning (ICML)*, pp. 6120–6130, 2021.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

Liu, G., Luo, Y., Schulte, O., and Poupart, P. Uncertainty-aware reinforcement learning for risk-sensitive player evaluation in sports game. In *Neural Information Processing Systems (NeurIPS)*, 2022.

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

Liu, S. and Zhu, M. Distributed inverse constrained reinforcement learning for multi-agent systems. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Liu, S. and Zhu, M. Learning multi-agent behaviors from distributed and streaming demonstrations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Liu, S. and Zhu, M. Meta inverse constrained reinforcement learning: Convergence guarantee and generalization analysis. In *International Conference on Learning Representations (ICRL)*, 2024.

Liu, Y., Halev, A., and Liu, X. Policy learning with constraints in model-free reinforcement learning: A survey. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4508–4515, 2021.

Ma, J. Y., Yan, J., Jayaraman, D., and Bastani, O. Offline goal-conditioned reinforcement learning via f-advantage regression. *Neural Information Processing Systems (NeurIPS)*, 35:310–323, 2022a.

Ma, Y., Shen, A., Jayaraman, D., and Bastani, O. Versatile offline imitation from observations and examples via regularized state-occupancy matching. In *International Conference on Machine Learning (ICML)*, pp. 14639–14663, 2022b.

Malik, S., Anwar, U., Aghasi, A., and Ahmed, A. Inverse constrained reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 7390–7399, 2021.

McPherson, D. L., Stocking, K. C., and Sastry, S. S. Maximum likelihood constraint inference from stochastic demonstrations. In *IEEE Conference on Control Technology and Applications, (CCTA)*, pp. 1208–1213, 2021.

Nachum, O., Chow, Y., Dai, B., and Li, L. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Neural Information Processing Systems (NeurIPS)*, pp. 2315–2325, 2019a.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019b.

Papadimitriou, D., Anwar, U., and Brown, D. S. Bayesian methods for constraint inference in reinforcement learning. *Transactions on Machine Learning Research*, 2023.

Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning (ICML)*, pp. 745–750, 2007.

Qiao, G., Liu, G., Poupart, P., and Xu, Z. Multi-modal inverse constrained reinforcement learning from a mixture of demonstrations. In *Neural Information Processing Systems (NeurIPS)*, 2023.

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

Sikchi, H., Zheng, Q., Zhang, A., and Niekum, S. Dual RL: Unification and new methods for reinforcement and imitation learning. In *International Conference on Learning Representations, ICLR*, 2024.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012.

Wang, X., Krasowski, H., and Althoff, M. Commonroad-rl: A configurable reinforcement learning environment for motion planning of autonomous vehicles. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 466–472. IEEE, 2021.

Xu, H., Jiang, L., Li, J., Yang, Z., Wang, Z., Chan, W. K. V., and Zhan, X. Offline RL with no OOD actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations, ICLR*, 2023.

Xu, S. and Liu, G. Uncertainty-aware constraint inference in inverse constrained reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2023.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. COMBO: conservative offline model-based policy optimization. In *Neural Information Processing Systems (NeurIPS)*, pp. 28954–28967, 2021.

Yue, S., Wang, G., Shao, W., Zhang, Z., Lin, S., Ren, J., and Zhang, J. CLARE: conservative model-based reward learning for offline inverse reinforcement learning. In *International Conference on Learning Representations, ICLR*, 2023.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. Modeling interaction via the principle of maximum causal entropy. In *International Conference on Machine Learning (ICML)*, pp. 1255–1262, 2010.

A. Implementation and Environment Details

A.1. Discrete Environments

Our discrete environments use a 7x7 gridworld map, with four different settings for constraint learning. In each setting, we assign a starting point where the agent is initially located and an absorbing point, which also functions as a terminate state. We employ value iteration to obtain the expert demonstration, and we shape the reward as $\hat{r} = r - 100c$ by subtracting a cost term to ensure that the demonstration is safe. For offline data collection, we sampled from random walks starting at grids [1,1], [1,5], [5,1], and [5,5]. To enrich the dataset with additional information, the data collect agent was instructed to choose a state it had not visited previously whenever making a move. To obtain the value function recovered by inverse imitation learning, we employed inverse q learning(Garg et al., 2021).

A.2. Continuous Environments

Our virtual environments are based on Mujoco. We utilize $c(s, a)$, derived from a state function $\mathbf{c}(s')$ representing state safety, with $c(s, a) = \mathbb{E}_{s' \sim P(s'|s,a)}[\mathbf{c}(s')]$. We provide more details about the virtual environments as follows:

1. **HalfCheetah(Obstacle)** This environments are taken from (Liu et al., 2023). In this environment, The agent controls a robot that can move faster backward than forward. The reward is based on the distance it moves between the current and previous time steps, along with a penalty related to the magnitude of the input action. We have defined a constraint that restricts movement in the region where the X-coordinate is less than or equal to -3, allowing the robot to move forward only.

2. **Ant (Blocked) and Hopper (Blocked)**

   In this environment, the agent controls a robot to move forward and obtain rewards related to moving distance. However, we enforce a constraint on the agent's leg angular velocity to prevent a large force on the ground. The limits are 1 for Ant and 0.3 for Hopper.

3. **Ant (LS) and Hopper (LS)**

   In this environment, the agent controls a robot to move forward and obtain rewards related to moving distance. However, it must not exceed a speed limit (0.5 for Ant and 1 for Hopper), resulting in obtaining fewer rewards compared to no constraint.

To obtain offline data, we first build a suboptimal dataset using two 200,000-step early stop SAC agents. We use the original reward and reward-shaping (modifying the reward by subtracting a cost) as reward signals and collect the replay buffers between 100,000 and 200,000 steps. Together, we obtain a suboptimal dataset with 200,000 steps.

For the expert dataset, we sample 50 trajectories from a PPO-lag algorithm. When sampling from the offline dataset, we maintain a balanced approach by selecting half of the samples from the suboptimal dataset and half from the expert trajectories.

A.3. Transfer Environments

In the Ant environment, we train an agent with the constraint $v_x < 1.0$ as the expert policy. For the transfer rewards, we modify the direction of the velocity in the x-axis in the original rewards v_x to be the velocity along a -45 degree angle, denoted as v_{-45°. For offline data, we employ PPO to train a policy that maximizes the transfer rewards, and PPO-lag to train a policy that maximizes rewards while adhering to the specified constraint. We sample 50 trajectories from each trained policy and merge together the data in D^O as offline data.

A.4. Incorporating Rewards in Inverse Reinforcement Learning

Since offline inverse reinforcement learning does not assume the existence of reward information, we construct a variant for inverse RL by modeling the reward model $\bar{r}$ in inverse RL with $\bar{r} = r + c$, where r is the ground truth reward function, and c is the cost function to be estimated. To be more specific, we incorporate the reward information in the following formula:

LS-IQ: We replace the original Q-learning objective (objective (20) in (Al-Hafez et al., 2023)) with

$$\mathcal{L}_{\text{lsiq-o}}(Q) = \alpha \mathbb{E}_{d_{\pi_E}}\left[\left(Q(s,\Gamma_\omega(s,s')) - Q_{\max} + r(s,a)\right)^2\right] + \bar{\alpha}\mathbb{E}_{d_\pi}\left[\left(Q(s,a) - (r_{\min} + r(s,a) + \gamma \mathbb{E}_{s' \sim P(.|s,a)}[V^*(s')])\right)^2\right], \tag{19}$$

SMODICE: We add the ground truth reward function SMODICE's learned discriminator reward $R(s,a)$ (objective (14) in (Ma et al., 2022b)) with

$$R_{\text{new}}(s,a) = R(s,a) + r(s,a) \tag{20}$$

A.5. Implementation Details

Across our set of environments, we maintain the same hyperparameters, except for the temperature parameter ξ_r. We select the parameter from the set [0.01, 0.001, 0.0001, 0.00001], choosing the one with the highest mean return from safe results (violation rate $<$ 0.05) and the lowest violation rate when all parameters are unsafe. We utilize a two-layer MLP with 256 hidden units and a ReLU activation function. The complete set of hyperparameters used in our experiments is presented in Table 5.

Hyperparameter	**Value**
Batch Size	256
Policy Learning Rate	3e-4
Value Learning Rate	3e-4
L2 Regularization Coefficient	0.0005
Random Seed α	1 2 3 4 5

Table 5: Hyperparameters for our experiments.

A.6. Experimental Equipment and Infrastructures

We ran the experiment on a cluster that has multiple RTX 3090 GPUs, each with 24 GB of memory. There is only one running node. With the aforementioned resources, running one seed in the virtual environment takes about 40 minutes.

B. Compared to Adversarial Imitation Learning

As proposed by (Ho & Ermon, 2016), the idea behind adversarial imitation learning is to learn a cost(reward) function to discriminate expert behavior from others:

$$\max_{r} \min_{\pi \in \Pi} L(\pi, r) = \mathbb{E}_{d^E}[c(s,a)] - \mathbb{E}_{d^\pi}[c(s,a)] - H(\pi)$$

In our ICRL setting, we can view our tasks as discriminating expert behavior from behavior within a smaller set, i.e. superior distributions. Because of information provided by predefined rewards, the feasible region of such max-min question is smaller and thus makes it become an easier problem to solve. This implies the potential for better policy recovery and a more transferable cost function.

C. Proof and Derivation

C.1. Deriving Lagrangian of Problem (6)

By introducing the Lagrange multiplier V and λ_d. The Lagrangian of problem (6) can be:

$$
\begin{aligned}
&\mathbb{E}_d[r(s,a)] - \xi_r D_f(d \,||\, d^O) - V(s)(\sum_{a\in\mathcal{A}} d(s,a) - (1-\gamma)\rho_0(s) - \gamma \sum_{(s',a')} d(s',a')p(s|s',a')) \\
=&\mathbb{E}_d\left[r(s,a) + \gamma \sum_{(s',a')} d(s',a')p(s|s',a'))V(s') - V(s)\right] + (1-\gamma)\mathbb{E}_{\rho_0}[V] - \xi_r D_f(d \,||\, d^O) \\
=&\mathbb{E}_d[\delta_V] - \xi_r D_f(d \,||\, d^O) + (1-\gamma)\mathbb{E}_{\rho_0}[V]
\end{aligned}
$$

C.2. Proof of Proposition 4.5

Proof. In order to solve:

$$
\begin{aligned}
&\max_d \ \mathbb{E}_d[\delta_V - c] - \xi_r D_f(d \,||\, d^O) + (1-\gamma)\mathbb{E}_{\rho_0}[V] && (21)\\
&\text{s.t. } \ d(s,a)c(s,a) \le 0 \text{ and } d(s,a) \ge 0 \ \forall s,a && (22)
\end{aligned}
$$

Assume $d(s,a) > 0$ implies $d^O(s,a) > 0$. We introduce a new variable $w(s,a) = \frac{d(s,a)}{d^O(s,a)}$ and redefine our optimization objective as follows:

$$
\begin{aligned}
&\max_w \ \mathbb{E}_{d^O}[w(s,a)\delta_V - \xi_r f(w(s,a))] + (1-\gamma)\mathbb{E}_{\rho_0}[V(s)] && (23)\\
&\text{s.t. } \ \omega(s,a)c(s,a) \le 0 \text{ and } \omega(s,a) \ge 0 && (24)
\end{aligned}
$$

Without the constraint on $d(s,a)$, the maximization of the first part is equivalent to finding the convex conjugate of the function f, denoted as $f^*(y) = \max_x(xy - f(y))$. However, we must consider the non-negativity constraint $d(s,a) \ge 0$ and feasible constraint $\omega(s,a)c(s,a) \le 0$.

We can form the Lagrangian function:

$$
\mathcal{L}(V,\lambda^c,\lambda) = \mathbb{E}_{d^O}[w(s,a)\delta_V - \xi_r f(w(s,a))] + (1-\gamma)\mathbb{E}_{\rho_0}[V(s)] + \sum_{s,a}(\lambda(s,a)\omega(s,a) - \lambda_c(s,a)c(s,a)\omega(s,a))
$$

The KKT condition of this problem is:

1.Primal feasibility $\omega(s,a)c(s,a) \le 0$, $\omega^*(s,a) \ge 0 \ \ \forall s,a$

2.Dual feasibility $\lambda_c^*(s,a) \ge 0$, $\lambda^*(s,a) \ge 0 \ \ \forall s,a$

3.Stationarity $\frac{\partial\mathcal{L}}{\partial V} = d^O(s,a)(-f'(\omega^*(s,a)) + \delta_V(s,a) + \lambda^*(s,a) - \lambda_c^*(s,a)c(s,a)) = 0 \ \ \forall s,a$

4.Complementary Slackness $\omega^*(s,a)\lambda^*(s,a) = 0$, $\lambda_c^*(s,a)\omega^*(s,a)c(s,a) = 0$

Using Complementary Slackness, we have two cases:

Case 1:$\lambda_c^*(s,a) = 0$, which means the solution without introducing constraint is feasible. Following a similar proof given in (Sikchi et al., 2024)(Refer to equation 43 in the appendix of the cited paper for more details), we solve the optimal solution with the stationary equation and first complementary slackness equation:

$$
\omega^*(s,a) = \omega^r(s,a) = \max(0, f'^{-1}(\frac{\delta_V(s,a)}{\xi_r}))
$$

. and $\omega^r(s,a)$ satisfies primal feasibility $\omega^*(s,a)c(s,a) = 0$.

Case2:$\lambda_c^*(s,a) \ne 0$, which means $\omega^*(s,a)c(s,a) = 0$. Which means we need to solve the following question. Note that such a solution is possible if only there exists some x s.t. $f'(x) = 0$.

$$\omega^*(s,a) = \max(0, f'^{-1}(\frac{\delta_V(s,a) - \lambda_c^*(s,a)c(s,a)}{\xi_r})) \tag{25}$$

$$\omega^*(s,a)c(s,a) = 0 \tag{26}$$

Summarizing these two case and replace ω^* with d^*, we have

$$d^*(s,a) = d^O(s,a)\mathbb{1}_{c(s,a)=0} \max(0, f'^{-1}(\frac{\delta_V(s,a)}{\xi_r})) \tag{27}$$

□

C.3. Proof of Proposition 4.2

Proof. ”→”: If $d = d^E$ solve the problem. then the $c(s,a) = 0$ for all $d^E(s,a) > 0$, otherwise the d^E is not feasible. And if there exists a $d \in O$ s.t. $d(s,a)c(s,a) = 0 \; \forall s,a$, then d is feasible and d is optimal than d^r in primal problem, which leads to a contradiction.

”←”: If $\forall d \in O$ there is at least one (s,a),s.t. $dc \neq 0$, then all $d \in O$ are all not feasible. Since $c(s,a) = 0 \; \forall (s,a) s.t. d^E(s,a) > 0$ d^E is feasible, so d^E is optimal.

□

C.4. Proposition of Maintaining Optimality

Proposition C.1. *If the learned cost function $c_k(s,a)$ satisfied the condition that $d^E(s,a) > 0 \implies c_k(s,a) = 0$, the distribution generated by the CRL problem (4) d^* is in superior distributions, i.e., $d^* \in \mathcal{O}$.*

Proof. If $d^* \notin \mathcal{O}$, than $\mathbb{E}_{d^*(s,a)}[r(s,a)] < \mathbb{E}_{d^E(s,a)}[r(s,a)]$. But since $d^E(s,a) > 0 \implies c_k(s,a) = 0$, d^E is a feasible solution for CRL problem (4), which contradicts to the optimality of d^*. □

Proposition C.2. *For a finite state space $\mathcal{S}$ and a finite action space $\mathcal{A}$, if d^* is the only optimal solution under the constraint, we initialize $c_0(s,a) = 0$. By alternately solving for d_i^* and updating c_i to project at any $d_i^*(s,a)$ such that at least $c^*(s,a)$ is set to 0 when $d_i^*(s,a) > 0$, while keeping $c_i(s,a) = 0$ for all $d^E(s,a) > 0$.*

Then, the algorithm will converge to d^E. During the learning process, the visiting distribution d_k^* maintains $J(d_k^*) \geq J(d^E)$.

Proof. We know $J(d) \geq J(d^E)$ since d^E is always a candidate for a solution in each round of the update. We now prove that if $d_{(k)}^* \neq d^E$, we can always find some s, a such that $d_{(k)}^*(s,a) > 0$ and $d^E(s,a) > 0$. First, we observe that for any policy d that visits fewer state-action pairs than d^E, i.e., if $\{(s,a) : d(s,a) > 0\} \subset \{(s,a) : d^E(s,a) > 0\}$, we have $J(d) < J(d^E)$. Otherwise, d is feasible, which would contradict the uniqueness of optimality of d^E. Thus, we can always find some (s,a) in $\{(s,a) : d(s,a) > 0\}$ but not in $\{(s,a) : d^E(s,a) > 0\}$.

From this, we also know that if the algorithm converges, it will always converge to d^E. We notice that for every round of updating, the number of zeros in $c_{(k)}(s,a)$ is increasing. Since in every round k, the optimal visitation distribution $d_{(k)}^*(s,a)$ with respect to $V_{(k)}^*$ satisfies $d_{(k)}^*(s,a)c_{(k-1)}(s,a) = 0$, and in round k, we must set some $c_{(k)}(s,a)$ to be above 0 for $d_{(k)}^*(s,a) > 0$.

Now, the spaces $\mathcal{S}$ and $\mathcal{A}$ are finite. We know the algorithm will converge to d^E since there are only a finite number of combinations of (s,a). □

D. Experiments

This section contains additional results.

D.1. Cost Visualization in HighD Dataset

We visualize the images with the highest assigned cost in each scenario using different methods. Notably, our method assigns higher costs to cars that are much closer in the test dataset, while OptiDICE-c failed to do so.

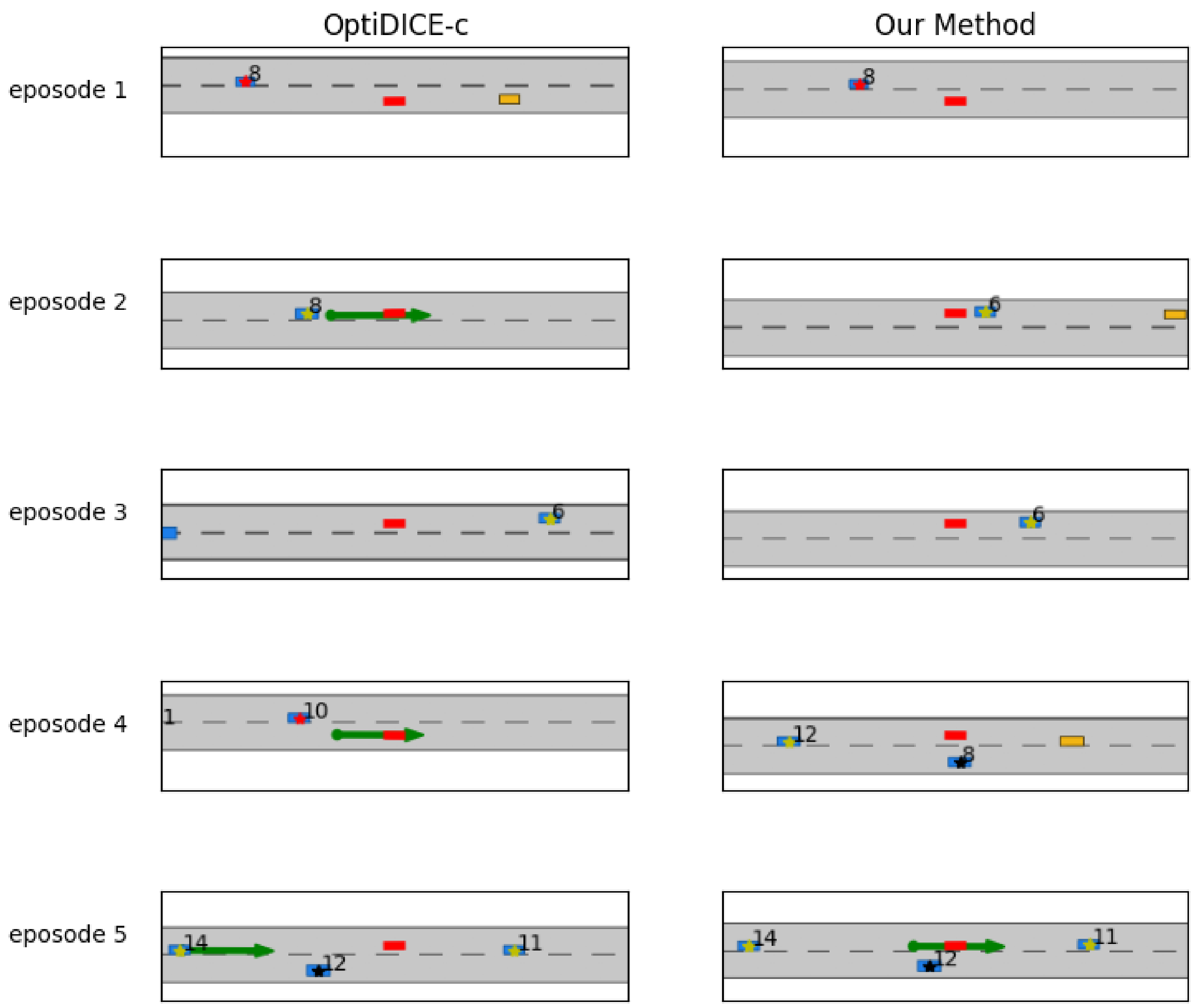

Figure 6: The red car represents the ego car, the blue cars are the other vehicles the agent is trying to maintain distance from, and the yellow block is the destination.

D.2. Training Curves for Baseline

D.3. Results in MuJoCo

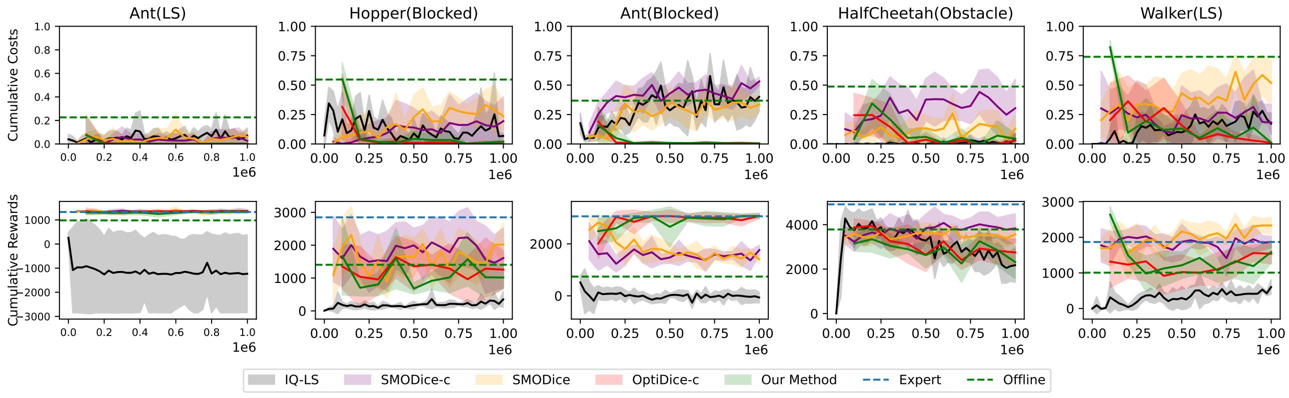

Figure 7: The cumulative rewards and the costs from the evaluation during training.

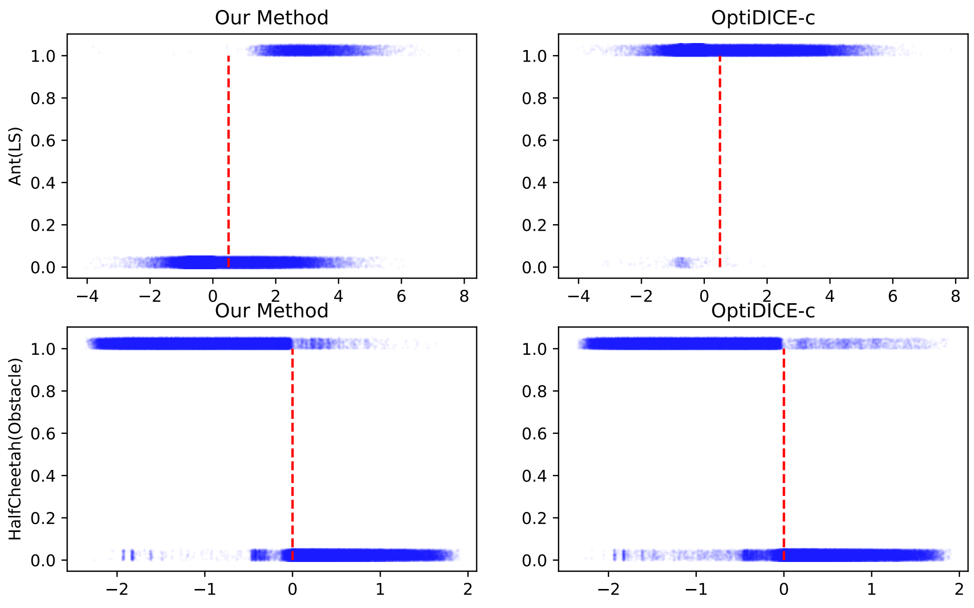

Figure 8: Recovered cost results: The x-axis represents the cost dimension of the state, and the y-axis represents the learned normalized cost. The plot is jittered along the y-axis for visualization.