Maximum Causal Entropy Inverse Constrained Reinforcement Learning

Mattijs Baert[1*], Pietro Mazzaglia[1], Sam Leroux[1], Pieter Simoens[1]

[1]IDLab, Ghent University-imec, Technologiepark-Zwijnaarde 126, Ghent, 9052, Belgium.

*Corresponding author(s). E-mail(s): mattijs.baert@ugent.be;

Abstract

When deploying artificial agents in real-world environments where they interact with humans, it is crucial that their behavior is aligned with the values, social norms or other requirements specific to that environment. However, many environments have implicit constraints that are difficult to specify and transfer to a learning agent. To address this challenge, we propose a novel method that utilizes the principle of maximum causal entropy to learn constraints and an optimal policy that adheres to these constraints, using demonstrations of agents that abide by the constraints. We prove convergence in a tabular setting and provide a practical implementation which scales to complex environments. We evaluate the effectiveness of the learned policy by assessing the reward received and the number of constraint violations, and we evaluate the learned cost function based on its transferability to other agents. Our method has been shown to outperform state-of-the-art approaches across a variety of tasks and environments, and it is able to handle problems with stochastic dynamics and a continuous state-action space.

Keywords: Inverse Constrained Reinforcement Learning, Principle of Maximum Causal Entropy, Constraint Inference, Constraint Learning, Safe Reinforcement Learning, Contrained Reinforcement Learning

1 Introduction

The behavior of individuals in today's society is shaped by social norms, which provide predictability, safety and efficiency in human interaction. Similarly, successful integration of artificial agents in the real-world requires *value alignment* of their behavioral

policies [1–3]. As an example, a self-driving car should not only efficiently transport its passengers to their destination, but it should also adhere to traffic regulations, such as traffic signs, traffic lights, and road conditions, to ensure the safety of all road users. Furthermore it is essential to program social norms into the agent prior to deployment to ensure that rule violations are avoided at all times. Learning value-aligned policies can be viewed as optimizing an objective subject to a set of constraints. In the framework of Reinforcement Learning (RL) this entails finding a policy that maximizes a reward function to which a weighted cost term was added that penalizes constraint violations. Not only does each combination of weights lead to a different Pareto optimal solution [4], also tuning them is a difficult and time-consuming process, as the task goal and the constraints often conflict with one another [5]. A more effective alternative is the use of Constrained Reinforcement Learning (CRL) methods which employ primal-dual methods on a constrained optimization problem, allowing to learn the optimal weights from data [6, 7]. Both RL and CRL methods require the manual specification of a cost function. Although it is straightforward to define constraints for simple environments, this is considerably harder for real-world settings, which often comprise multiple and implicit constraints. Returning to the traffic example, human drivers also adopt implicit rules, for example driving slower when traffic is heavy or leaving more space when driving behind a truck. To address this problem, we advocate for learning a cost function that encapsulates the environment's constraints from demonstrations provided by constraint-abiding agents, a paradigm known as Inverse Constrained Reinforcement Learning (ICRL).

Recent advancements in Inverse Reinforcement Learning (IRL) have facilitated the acquisition of reward functions from expert demonstrations in challenging environments [8–10]. However, there has been comparatively limited exploration in the realm of learning cost functions (i.e. ICRL). It is essential to note that in ICRL, the assumption is that the (goal-oriented) reward function of the expert is available, and the focus is solely on learning the constraints. Despite the apparent similarity between IRL and ICRL, a key distinction lies in how they handle states absent in expert demonstrations. The unvisited states, those not encountered by the expert, can be categorized into a group of constrained states and a group of states which are unconstrained but are associated with low reward potential. IRL fails to distinguish between these two groups of unvisited states, potentially leading to constraint violations if the agent explores states the expert did not visit. In contrast, ICRL explicitly separates these two groups by imposing high costs on constrained states, ensuring that the agent consistently avoids constrained states.

In this work, we propose a novel constrained optimization objective for addressing the ICRL problem. Unlike previous approaches [11–15], our methodology seamlessly scales to environments characterized by a continuous state-action space with unknown and stochastic transition dynamics. To avoid the potential issue of overfitting the constraints on the expert data, we adopt the maximum entropy principle [16]. This principle facilitates the learning of constraints that align with expert data while minimizing bias. In contrast to conventional methods [11–14, 17–19], we adopt an alternative entropy definition that respects the causality inherent in states and actions within trajectories [20]. This distinction enables our method to be effectively applied

to environments characterized by stochastic transition dynamics [1].
Our primary contributions encompass: i) the formulation of a novel ICRL objective grounded in the principle of maximum causal entropy, ii) the derivation of an exact solution for the proposed objective, supported by theoretical proofs, and iii) the development of an approximate method that extends the applicability of our approach to environments characterized by continuous state-action spaces. Our method simultaneously learns a cost function aligned with expert trajectories and a policy that maximizes the given reward function while minimizing the learned cost function. We evaluate the performance of the learned policy in terms of received rewards and the number of constraint violations across virtual and realistic environments. We explore the impact of varying levels of stochasticity in the environment's dynamics on these performance metrics. Additionally, we assess the transferability of the learned cost function to different types of agents with distinct reward functions. Lastly, we conduct an ablation study to investigate the importance of different components within our proposed method. Empirical evaluation indicates that our method consistently outperforms state-of-the-art techniques across various benchmarks.
The structure of the paper is as follows: In Section 2, we discuss related work. In Section 3, we present the necessary background information on constrained Markov decision processes and inverse reinforcement learning. Our proposed method is outlined in Section 4. We evaluate and compare the results of our method to existing state-of-the-art approaches in Section 5. Finally, we conclude and provide future directions in Section 6.

2 Related Work

2.1 Inverse Constrained Reinforcement Learning

A substantial body of current research aims to discern the set of constraints, or cost function, which maximizes the likelihood of the expert trajectories. This maximum likelihood objective can be solved by employing methodologies such as greedy algorithms [11–14] or gradient-based methods [17–19, 21]. Kim et al. [22] formulate the ICRL objective as a zero-sum game between a policy player and a constraint player. Similarly to the previous methods, solving this objective yields a set of constraints corresponding to the states with high reward potential that were not visited by the expert. Opposed to our approach, these methods only consider environments with deterministic transition dynamics limiting their applicability in real-world scenarios.
An alternative line of work treats constraint learning as a one-class classification problem [23]. In this study, the central goal is to develop a model of expert behavior. Next, regions in the state-action space which are unlikely under the expert model are designated as constraints. In the same line of work, Lindner et al. [24] employ a convex hull algorithm to obtain a safe set from the expert feature expectations. One major drawback of these methods is their vulnerability against overfitting on the expert data. In contrast, we adopt entropy regularization to learn an unbiased cost function. Papadimitriou et al. [15] offer a distinctive perspective by learning a distribution over potential

[1] Please refer to Sec. A in the appendix for a comprehensive explanation on why standard entropy is inappropriate in scenarios where stochastic transition dynamics are present.

constraints. Although, this Bayesian approach introduces a nuanced understanding of uncertainty in the constraint inference process, it is limited to simple environments.
While some prior work [17, 25], considers soft constraints, the majority of methods concentrate on learning hard constraints [11–14, 19, 22, 24]. In the context of reinforcement learning, hard constraints must always be satisfied, whereas soft constraints can be violated as long as they are satisfied on average (i.e., in expectation). In theory, our method supports both types of constraints. However, in this work, we focus on hard constraints, and, as of now, our experiments have exclusively involved hard constraints.
Opposed to our method, several approaches [11–15] necessitate the transition probability $p(s' \mid s, a)$ can be explicitly defined. Additionally, these approaches require multiple iterations over the entire state space. As a consequence, their applicability is confined to discrete environments characterized by a limited state space.
The constraint learning problem is inherently ill-defined, as many different sets of constraints can explain the same expert trajectories. In the extreme case, all states not visited by the expert, might be considered as constraints [24]. To recover the smallest set of constraints which explain the expert trajectories, many methods [11–14, 17–19] adopt the principle of maximum entropy [16]. All these methods, unlike ours, rely on the standard entropy definition, rendering them unsuitable for environments with stochastic transition dynamics. By contrast, the approach proposed by McPherson et al. [13] also employs maximum causal entropy, enabling constraint learning in stochastic environments. However, their method necessitates enumerating the entire constraint set, which makes it impractical not only for continuous state-action spaces but also for large discrete spaces. In a different approach, Kim et al. [22] leverage multi-task demonstrations as a strategy to mitigate overfitting to expert demonstrations. However, this strategy intensifies the dependence on the availability of diverse expert demonstrations and the existence of multiple tasks within the environment.

2.2 Learning From Experts

A variety of methods have leveraged human expertise to improve the performance of autonomous agents. Behavioral cloning relies on expert demonstrations to learn a policy through supervised learning [26, 27]. Despite its simplicity and efficiency in specific application contexts, it is possible there will be a mismatch between the training and testing state distribution, leading to a notable decline in performance. In preference learning, the goal is to learn a mapping between inputs and a ranking of those inputs which reflects the user's (or expert's) preferences [28, 29]. While humans often find it more straightforward to convey preferences rather than providing complete demonstrations, the latter conveys a richer set of information. There are other works from the robotics community that focus on learning constraints from demonstrations. In [30], given the environment's dynamics, a set of unsafe trajectories is sampled, and constraints are learned by solving an integer programming problem using both the unsafe trajectories and the set of demonstrations. In [31], the scalability of this method is improved by exploiting constraint parameterizations. Further, in [32], they extend their method to accommodate locally optimal demonstrations, and in [33, 34], they address how constraint uncertainty can be managed. However, their approach

assumes that goal-satisfying trajectories can be easily sampled. This is a requirement that becomes impractical for tasks involving long control sequences within large or continuous action spaces. Most closely related to ICRL is IRL which has the goal of learning and subsequently optimizing a reward function [8, 35]. However as we will demonstrate in the experiment section, IRL methods prove unsuitable for learning constraints.

3 Background

3.1 Constrained Markov Decision Process

A Markov decision process (MDP) is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a discount factor $\gamma \in [0, 1]$, a transition distribution $p(s' \mid s, a)$ which specifies the probability of transitioning to state s' when performing action a while in state s, an initial state distribution $\mathcal{I}(s)$ and a bounded reward function $R : \mathcal{S} \times \mathcal{A} \mapsto [r_{\min}, r_{\max}]$ specifying the scalar reward the agent receives for applying action a while in state s. We consider an agent interacting with the environment at discrete timesteps t generating a sequence of transitions called a trajectory $\tau = ((s_0, a_0, s_1), ..., (s_{T-1}, a_{T-1}, s_T))$ of length T. We define the reward of a trajectory as the sum of discounted rewards: $R(\tau) = \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)$. At every timestep, the agent selects its action based on a policy $\pi : \mathcal{S} \mapsto \mathcal{P}(\mathcal{A})$ which is a mapping from a state to a probability distribution over actions. The goal of forward reinforcement learning is to find the policy which maximizes the expected sum of discounted rewards: $\max_\pi \mathbb{E}_{\tau\sim\pi} R(\tau)$.

A constrained Markov decision process (CMDP) $\mathcal{M}^C$ [36] is defined as an MDP augmented with a non-negative bounded cost function $C : \mathcal{S} \times \mathcal{A} \mapsto [c_{\min}, c_{\max}]$ and a budget $\alpha \geq 0$. $C(s, a)$ denotes the cost of taking action a in state s and the cost of a trajectory is defined as the sum of discounted costs: $C(\tau) = \sum_{t=0}^{T-1} \gamma^t C(s_t, a_t)$. The goal of constrained reinforcement learning is defined as to find the policy which maximizes the expected sum of discounted rewards while the expected sum of discounted costs is smaller than the budget:

$$\max_\pi \mathbb{E}_{\tau\sim\pi} R(\tau) \;\text{ s.t. }\; \mathbb{E}_{\tau\sim\pi} C(\tau) < \alpha. \tag{1}$$

When α is 0, constraints can be interpreted as hard constraints meaning the resulting policy should never visit states which incur any cost. When $\alpha > 0$, the resulting policy is allowed to obtain cost > 0 in some cases, i.e. soft constraints.

3.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is a learning problem which, given an expert data distribution $\mathcal{D}$ represented by a finite set of trajectories, tries to identify the reward function $R(s, a)$ the expert agent is optimizing. Assume the unknown reward function is defined as $R'(s, a) = \omega \cdot \phi(s, a)$ with $\omega \in \mathbb{R}^k$ a k-dimensional vector of real numbers and $\phi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_+^k$ denoting a fixed mapping from the state-action space to a k-dimensional vector of non-negative features. We define the feature representation of a trajectory as the sum of discounted feature vectors: $\phi(\tau) = \sum_{t=0}^{T-1} \gamma^t \phi(s_t, a_t)$.

With forward RL we can obtain a policy which maximizes R' (i.e. the nominal policy). The problem of IRL can then be rephrased as finding values of ω such that the expected features when following the nominal policy $\mathbb{E}_{\tau\sim\pi}[\phi(\tau)]$ match the expected features under the expert data distribution $\mathbb{E}_{\tau\sim\mathcal{D}}[\phi(\tau)]$, i.e. feature expectation matching [35]. However, this is an ill-posed problem as many assignments of ω will result in matching expected features. To resolve this ambiguity, Ziebart et al. [20] proposed to choose the values ω that result in matching feature expectation and correspond to the nominal policy that maximizes the causal entropy, i.e. Maximum Causal Entropy Inverse Reinforcement Learning (MCE-IRL). Under Markovian dynamics the causal entropy of a trajectory is defined as the discounted sum of the conditional entropy of the current action given the current state: $H(\tau) = \sum_{t=0}^{T-1} \gamma^t H(a_t \mid s_t)$. Although the original feature matching objective only considers reward functions that are defined as a linear combination of individual dimensions of ϕ, state-of-the-art methods are scalable to complex problems by parameterizing R' with a deep neural network [8, 37]. For an elaborate introduction to MCE-IRL we refer the reader to the work of Gleave and Toyer [38].

4 Maximum Causal Entropy Inverse Constrained Reinforcement Learning

In this section, we present Maximum Causal Entropy Inverse Constrained Reinforcement Learning (MCE-ICRL). First, we formalize our objective in Section 4.1. In Section 4.2, we present a detailed description of MCE-ICRL. To conclude, we provide a theoretical analysis in Section 4.3.

4.1 Problem Formulation

Inverse Constrained Reinforcement Learning (ICRL) methods aim to learn a cost function $C(s, a)$ that characterizes the constraints inherent in a specific environment. This is achieved by leveraging demonstrations $\mathcal{D}$ from agents that adhere to these constraints, and a known reward function $R(s, a)$ that defines their objectives. We will adopt the IRL terminology and will refer to constraint-abiding agents as expert agents. We define the following requirements for our novel ICRL method. (i) Our method should prevent overfitting on the expert data. (ii) Our method should scale to environments with a continuous state space. (iii) Our method should be applicable to scenarios with stochastic transition dynamics.

4.2 Algorithm

Although the goal of ICRL is to learn a cost function $C(s, a)$, we define the primal objective as identifying a stochastic policy $\pi(a \mid s)$, referred to as the nominal policy. As we will demonstrate in the following sections, solving this objective will enable us to also extract a cost function. We define the optimal nominal policy as the policy which maximizes the provided reward signal R and the causal entropy H, while concurrently

aligning its feature expectations with the expert trajectories:

$$\begin{aligned} &\max_{\pi \in \Pi} \quad \mathbb{E}_{\tau \sim \pi}\left[R(\tau) + \beta H(\tau)\right] \quad \text{s.t.} \\ &\mid \mathbb{E}_{\tau \sim \mathcal{D}}\left[\phi(\tau)\right] - \mathbb{E}_{\tau \sim \pi}\left[\phi(\tau)\right] \mid \leq \boldsymbol{\alpha}_k. \end{aligned} \tag{2}$$

With $\boldsymbol{\alpha}_k = (\alpha, \alpha, ..., \alpha) \in \mathbb{R}^k$ and where Π denotes the set of normalized policies with non-negative probabilities for all states and actions:

$$\pi \in \Pi \quad \Leftrightarrow \quad \pi(a \mid s) \geq 0 \;\text{ and }\; \int \pi(a \mid s)\, \mathrm{d}a = 1 \quad (\forall s \in \mathcal{S}). \tag{3}$$

We introduce the entropy coefficient β to strike a balance between the reward and entropy objectives. The alignment of feature expectations between the nominal policy and the expert data is enforced by imposing an inequality constraint with a specified budget α. To address the constrained optimization problem in eq. (2), we define the Lagrangian and determine its saddle points by solving the associated dual problem. The Lagrangian for the primal problem is obtained by replacing the feature matching constraint with a weighted penalty term in the objective, with weights $\lambda \in \mathbb{R}^k$ (i.e., the dual variable vector). Adhering to the Karush-Kuhn-Tucker (KKT) conditions, which stipulate that for any pair of optimal points (π, λ), all values of λ should be non-negative (Sec. 5.5.3 in Boyd et al. [39]), the Lagrangian of the primal problem (eq. (2)) is expressed as:

$$\mathcal{L}(\pi, \lambda) = \mathbb{E}_{\tau \sim \pi}\left[R(\tau) + \beta H(\tau)\right] + \lambda \cdot \left(\mathbb{E}_{\tau \sim \mathcal{D}}\left[\phi(\tau)\right] - \mathbb{E}_{\tau \sim \pi}\left[\phi(\tau)\right] - \boldsymbol{\alpha}_k\right). \tag{4}$$

Then the dual problem can be defined as

$$\min_{\lambda} \max_{\pi \in \Pi} \mathcal{L}(\pi, \lambda). \tag{5}$$

The Lagrange multiplier functions to achieve equilibrium between maximizing reward and ensuring the agent's behavior aligns with that of the expert. In the MCE-ICRL framework, two sequential stages tackle the optimization process, alternating between refining the policy according to Equation (4) and adjusting the dual variables λ to optimize the same equation. Intuitively, it's logical to regard behaviors yielding high rewards but absent in the expert's trajectories as potential constraints. There likely exists a governing principle preventing the expert from executing this high-reward behavior. Consequently, when the nominal policy deviates significantly from the expert's, the values of λ will increase. This increase ensures that the penalties for deviating from these conceivable constraints become more pronounced, reinforcing alignment with the expert's behavior. As demonstrated in Section 4.3, solving the dual problem outlined in equation (5) not only enables us to learn a policy that adheres to the constraints of the environment but also allows us to derive a cost function $C(s, a) = \lambda \cdot \phi(s, a)$ which defines the constraints of the environment. Algorithm 1 provides an overview of the proposed approach. This algorithm fulfills all our predetermined requirements: (i) Incorporating entropy into our objective ensures that

the learned policy remains unbiased. (ii) The gradient of the Lagrangian in Equation (4) can be computed w.r.t. the policy parameters, facilitating the utilization of highly scalable policy gradient reinforcement learning methods. (iii) By embracing causal entropy, we acknowledge the causal relationships between states in an MDP, thus accommodating stochastic transition dynamics (see Appendix A).

Algorithm 1 MCE-ICRL algorithm

Input: reward function $R(s, a)$, expert demonstrations $\mathcal{D}$
Output: cost function $C(s, a) = \lambda \cdot \phi(s, a)$, nominal policy π_θ
Parameter: number of iterations η

1: Initialize θ, λ and ζ
2: Learn nominal policy π_θ^0 with zero cost
3: Pre-train ϕ_ζ using $\tau \sim \pi_\theta^0$ and $\tau \sim \mathcal{D}$ (Sec. 4.3.2)
4: **for** $i \leftarrow 0$ **to** η **do**
5: Learn π_θ maximizing eq. (4) w.r.t. to the policy parameters θ (Sec. 4.3.1, eq. (11)).
6: Perform gradient descent on λ and ζ minimizing eq. (4) w.r.t. λ and ζ (Sec. 4.3.2, eq. (12) and eq. (13)).
7: **end for**

4.3 Algorithm Analysis

Theorem 1. *The dual problem in eq. (5) can be solved by alternately optimizing for π and λ. When the policy update is performed on a faster timescale than the updates on the dual variable vector, π and λ will converge to a local optimum. Proof. See chapter 6 of [40].*

Following Theorem 1, we iteratively solve the dual problem by optimizing the policy π (Sec. 4.3.1) and the dual variables λ (Sec. 4.3.2) alternately. Also in Sec. 4.3.2, we show we can replace the hard-coded feature representation ϕ with a neural network ϕ_ζ.

4.3.1 Policy Optimization

The maximization of the Lagrangian with respect to the policy corresponds with solving a planning problem, thereby rendering it amenable to RL techniques.

Proposition 2. *The optimal policy π^* maximizing eq. (4) is given by*

$$\pi^*(a_t \mid s_t) = \exp\left(\frac{1}{\beta}\big(Q^*(s_t, a_t) - V^*(s_t)\big)\right). \tag{6}$$

With the action-value function defined by

$$Q(s_t, a_t) = R(s_t, a_t) - \lambda \cdot \phi(s_t, a_t) + \gamma\, \mathbb{E}_p\left[V(s_{t+1})\right], \tag{7}$$

and the state-value function by

$$V(s_t) = \beta \log \int \exp\left(\frac{1}{\beta} Q(s_t, a)\right) \mathrm{d}a. \tag{8}$$

Proof. See Section C.1 in the appendix.

Proposition 2 provides expressions for the optimal policy π^* (eq. (6)), the action value function Q (eq. (7)), and the state value function V (eq. (8)). The form of this action value function closely resembles the one found in the standard entropy-regularized RL objective [41]. It includes an additional term that subtracts $\lambda \cdot \phi(s, a)$ from the reward. This augmentation, originating from the feature matching objective as demonstrated in the appendix (Sec. C.1), can be interpreted as a cost subtracted from the reward. According to theorem 3, the optimal policy can be derived through an iterative process involving policy evaluation and policy improvement. Policy evaluation entails inferring the action value function from any fixed policy, accomplished by iteratively applying a modified Bellman backup operator $\mathcal{T}^\pi$ given by

$$\mathcal{T}^\pi Q(s_t, a_t) \triangleq R(s_t, a_t) - \lambda \cdot \phi(s_t, a_t) + \gamma \, \mathbb{E}_p \left[V(s_{t+1}) \right], \tag{9}$$

with $V(s) = \mathbb{E}_\pi \left[Q(s, a) - \beta \log \pi(a \mid s) \right]$ (derived from eq. (6)). During the subsequent policy improvement step, the policy for each state is updated toward the optimal policy using the computed value functions:

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}} \left(\pi'(\cdot \mid s_t) \,\middle\|\, \exp\left(\frac{1}{\beta}\big(Q(s_t, a_t) - V(s_t)\big)\right) \right). \tag{10}$$

Theorem 3 (Policy Iteration)**.** *Continually applying policy evaluation (using eq. (9)) and policy improvement (using eq. (10)) starting from any $\pi \in \Pi$, converges to the optimal policy π^* for which $Q^{\pi^*}(s_t, a_t) \geq Q^\pi(s_t, a_t)$ for all $\pi \in \Pi$ and $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, assuming $|\mathcal{A}| < \infty$. Proof. See Section C.2 in the appendix.*

Policy iteration (theorem 3), however, is confined to a tabular setting with a limited state-action space. Recognizing this limitation, we introduce a practical algorithm designed to scale to continuous state-action spaces. In this context, we parameterize the policy using a neural network with parameters denoted as θ and adopt a policy gradient algorithm to attain the optimal policy.

Proposition 4. *Suppose that the policy π_θ is differentiable with respect to its parameters θ. Then*

$$\nabla_\theta \mathcal{L}(\theta, \lambda) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right.$$

$$\Big(Q(s_t, a_t) - \beta \log \pi_\theta(a_t \mid s_t) - \sum_{t'=t}^{T-1} \gamma^{t'-t}\beta\Big)\Big]. \quad (11)$$

Proof. See Section C.4 in the appendix.

The gradient in equation (11) closely resembles the standard policy gradient [42]. However, the maximum entropy objective introduces the additional term: $-\beta \log \pi_\theta(a_t \mid s_t) - \sum_{t'=t}^{T-1} \gamma^{t'-t}\beta$.

4.3.2 Optimization of Dual Variables

Proposition 5. *The Lagrange dual problem (eq. (5)) is a convex optimization problem. Proof. See Section C.3 in the appendix.*

Because of proposition 5, we can use gradient descent to optimize the dual variables such that they converge to a global minimum. Then, theorem 1 and 3 and proposition 5 prove convergence for algorithm 1 in the tabular setting.
The step we need to take on the dual variable vector is obtained by taking the gradient of the Lagrangian with respect to λ:

$$\nabla_\lambda \mathcal{L}(\theta, \lambda) = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\phi(\tau)\right] - \mathbb{E}_{\tau\sim\pi_\theta}\left[\phi(\tau)\right] - \boldsymbol{\alpha}_k. \quad (12)$$

Following each update, any negative values of λ are constrained to zero, as the KKT conditions necessitate that all values of λ should be non-negative. Feature expectations under the nominal policy are computed by iterating over a set of trajectories sampled from the learned policy. Conversely, for the feature expectations under the expert policy, the iteration involves the set of provided expert trajectories. In an intuitive sense, if we interpret $\lambda \cdot \phi(s, a)$ as a cost function $C(s, a)$ in equation (7), the update to λ is designed such that features occurring in trajectories sampled from the nominal policy but not present in the expert data result in a higher cost. This feature matching enforces adherence to the expert data during the learning process.
Because the cost function is defined as the dot product of the transposed Lagrange multiplier λ and the feature representation $\phi(s, a)$, our method is limited to cost functions which are linear with respect to ϕ. In order to learn non-linear cost functions, we replace the hard-coded feature representation with a neural network with parameters ζ. We treat ζ as one of the dual variables and thus update them together with λ. To do this we derive the gradient of the Lagrangian w.r.t. ζ:

$$\nabla_\zeta \mathcal{L}(\theta, \lambda, \zeta) = \lambda \cdot \left(\mathbb{E}_{\tau\sim\mathcal{D}}\left[\nabla_\zeta \phi_\zeta(\tau)\right] - \mathbb{E}_{\tau\sim\pi_\theta}\left[\nabla_\zeta \phi_\zeta(\tau)\right]\right). \quad (13)$$

The outputs of ϕ_ζ are passed through a sigmoid activation function to assure positive feature values. We obtained better results by pre-training the feature encoder network ϕ_ζ. To do this, we define a feature decoder network and train the encoder and decoder as an autoencoder trying to minimize the reconstruction error between the original data (i.e. encoder input) and the reconstruction (i.e. decoder output). The autoencoder is trained on nominal and expert data.

5 Experiments

We conducted a comprehensive validation of our approach across various settings, including a gridworld environment and the ICRL benchmark proposed by Liu et al. [19]. This benchmark encompasses five robotic domains and a realistic traffic environment. For our experiments, expert trajectories were sampled from a policy trained using reward-constrained policy optimization [7], adhering to ground truth constraints. Throughout the experiments, we employed the approximate policy gradient method to optimize the policy (Sec. 4.3.1). Specifically, the nominal policy was derived using the policy gradient method, Proximal Policy Optimization (PPO) [43]. Our set of validation experiments includes a study on the transferability of the learned cost function to other agents, an ablation study on the pre-training of the feature encoder, and an investigation into the influence of the hyperparameter β. Detailed information on hyperparameters and experimental settings can be found in Appendix E. To assess the effectiveness of our method, denoted as MCE-ICRL, we compared it against three baseline methods:

- **Generative Adversarial Constraint Learning (GACL)**: This method is based on the well-established imitation learning method: generative adversarial imitation learning (GAIL) [8]. Following Malik et al. [18], GAIL can be adopted for ICRL by training the discriminator $D(s, a)$ to discriminate expert from nominal trajectories. The discriminator is trained such that, $D(s, a)$ will output values close to 1 for state-action pairs drawn from expert trajectories and values close to zero for state-action pairs which only occur in the nominal trajectories (i.e. possible constrained state-action pairs). During the forward step, the policy is trained by maximizing $\bar{r}(s, a) = r(s, a) + \log D(s, a)$. Because $\bar{r}$ becomes $-\inf$ for constrained state-action pairs, the agent will try to satisfy all constraints.
- **Maximum Entropy Inverse Constrained Reinforcement Learning (ME-ICRL)** [18]: This method adopts the principle of maximum entropy [16] for modeling the likelihood of trajectories.
- **Variational Maximum Entropy Inverse Constrained Reinforcement Learning (VME-ICRL)** [19]: This method also builds on the principle of maximum entropy but models constraints as a beta distribution.
- **Learning Constraints from Demonstrations (LCfD)** [31] : This method uses hit-and-run sampling to generate unsafe (i.e. nominal) trajectories. By solving an integer program, a representation of the constraints can be learned.

We intentionally chose not to compare our method with approaches that assume knowledge of the transition function and require exhaustive iteration over the entire state-action space [11–14]. Such a comparison would be unfair, as those methods assume complete knowledge of transition dynamics, whereas our approach only requires a simulator to sample trajectories based on a given policy.
We evaluate the nominal policy at different points during training by sampling trajectories from it and reporting the average obtained reward and the average constraint violation rate. The reward denotes the total reward an agent has received during a trajectory. The constraint violation rate is the fraction of all timesteps of a trajectory that violate at least one constraint. In all experiments, we consider hard constraints

by setting the budget α to zero. This ensures that the feature expectations of the nominal and expert policy align (Eq.(2)).

5.1 Gridworld

We designed a gridworld environment where the agent's goal is to travel from a fixed initial location to a fixed goal location while avoiding constrained locations. A screenshot of this environment is shown in Sec. D in the appendix (Figure D1). Every step the agent receives a reward of -1 except when it reaches the goal state, the agent gets a reward of 1. Each episode lasts a maximum of 200 timesteps. We use this environment to examine the influence of stochastic environment dynamics on the performance of learning a cost function and retrieving the optimal policy. With a particular probability, the environment ignores the action taken by the agent and instead transitions it to a randomly chosen neighboring cell. We call this probability the stochasticity of the environment. Figure 1 shows the obtained reward and constraint violations at test time for increasing stochasticity. We observed a small decrease of reward and small increase of constraint violations for MCE-ICRL for increasing stochasticity. Non-causal entropy approximates causal entropy for small stochasticity rates but fails when the randomness increases, this is reflected in the results of ME-ICRL and VME-ICRL. GACL consistently attains the highest rewards, albeit at the highest cost. The presence of scenarios wherein substantial constraint violations coincide with these elevated rewards raise concerns about the restrictiveness of the cost function. We conjecture that the discriminator may encounter challenges in generalizing the observed violated constraints in nominal trajectories to unseen state-action pairs. These results support our claim that IRL methods are not suitable for learning constraints. LCfD assumes that trajectories meeting start and goal constraints can be readily sampled. To narrow the sample space, a minimum reward threshold is set based on the average reward achieved in expert demonstrations. This approach works flawlessly in deterministic environments, consistently recovering the ground truth constraints. However, as stochasticity increases, LCfD fails to consistently recover the true constraints. To address stochastic dynamics, we introduce LCfD+, which lowers the reward threshold to include less optimal trajectories in the sampling process. This adjustment reduces the rate of constraint violations compared to LCfD. However, lowering the reward threshold expands the trajectory sampling space, leading to increased computational complexity, particularly when dynamics are unknown. Consequently, we find this method impractical for more complex tasks when dynamics are unknown. In Sec. F.1 of the appendix, we report extended results of experiments in the gridworld environment.

5.2 Virtual Robotics Environments

The virtual robotic environments are based on the MuJoCo environments from OpenAI Gym [44] but augmented with the constraints originally proposed by Malik et al. [18] and extended by Liu et al. [19]. We evaluate on five simulated robots: ant, halfcheetah, swimmer and inverted pendulum. Figure D2 (Sec. D.2 in the appendix) depicts a screenshot of each environment. This environment is characterized by a continuous state-action space and deterministic dynamics. The reward of the ant agent

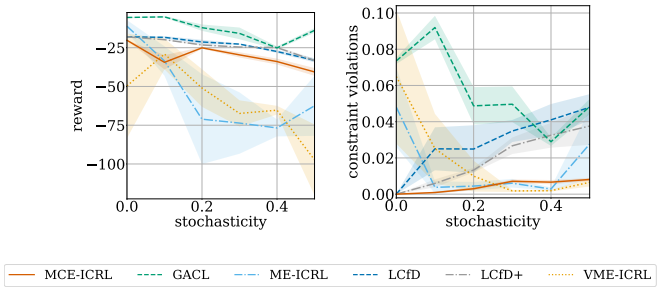
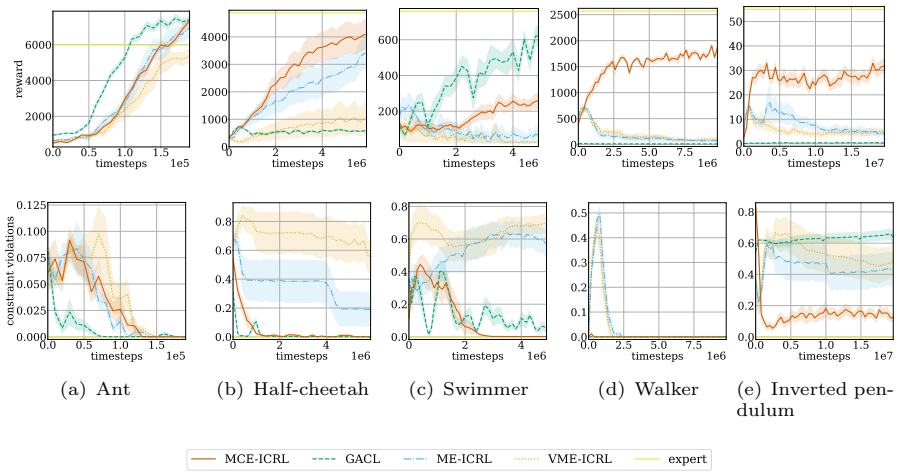


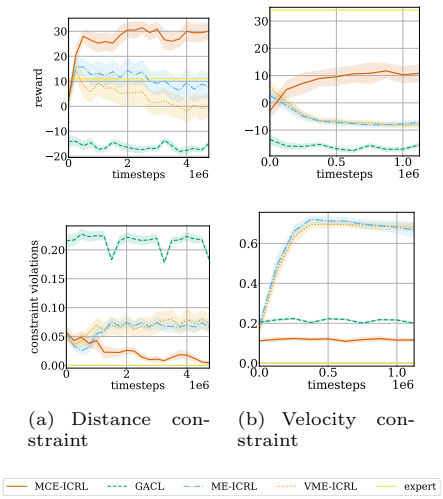
Fig. 1 Evaluation of the different methods in the gridworld environment for increasing stochasticity: reward (left) and constraint violation rate (right) of trajectories sampled from the nominal policy after training. Results are averaged over 5 random seeds. The x-axis corresponds with the stochasticity. The shaded regions correspond to the standard error.



(a) Ant (b) Half-cheetah (c) Swimmer (d) Walker (e) Inverted pendulum

Fig. 2 Evaluation of the different methods in the virtual robotics environments: reward (top) and constraint violation rate (bottom) of trajectories sampled from the nominal policy during training. Results are averaged over 10 random seeds. The x-axis is the number of timesteps taken in the environment during training. The shaded regions correspond with the standard error.

is proportional to the distance traveled from the starting position. The reward of the half-cheetah, swimmer and walker agents is determined by the distance it moves each step. Because of their morphology, each of these agents can move "easier" in one direction than in all other directions, but the ground truth constraints invalidate moving in the "easy" direction. The agent controlling the inverted pendulum receives higher rewards when it is to the left of the starting position. However, the ground truth constraints correspond to these high reward locations. The agent should learn to balance the pendulum to the right from the origin receiving lower rewards but also lower costs. Results are depicted in figure 2. In the ant environment MCE-ICRL performs comparable to the other methods. In the half-cheetah, walker and inverted pendulum domain our method results in higher rewards and less constraint violations. In the swimmer



(a) Distance constraint

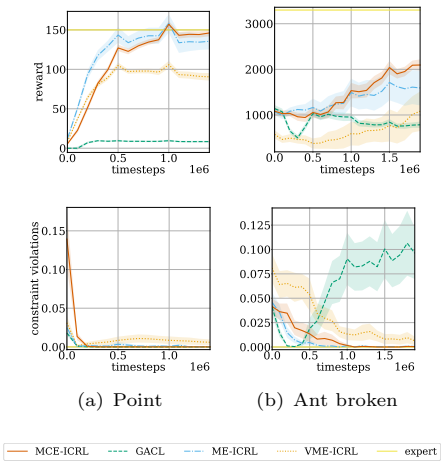
(b) Velocity constraint

Fig. 3 Comparison of our method against various baselines in the realistic traffic environment, with distance constraints (left) and velocity constraints (right). The top two plots showcase the obtained rewards, while the bottom two plots depict the constraint violation rates of trajectories sampled from the nominal policy during training. The results are averaged over 10 random seeds, with the x-axis representing the number of timesteps taken in the environment during training. The shaded regions correspond to the standard error.

environment MCE-ICRL results in the lowest cost and higher rewards than ME-ICRL and VME-ICRL. GACL obtains higher rewards but with more constraint violations. For every task, we also plot the reward and constraint violation rate obtained during the expert trajectories. Note there is often still a gap between the performance of the learning agent and the expert.

5.3 Realistic Traffic Environment

The realistic traffic environment comprises a highway driving task. This task was proposed by Liu et al. [19]. This environment is constructed from the highD dataset [45] which is a dataset of naturalistic trajectories of vehicles on German highways. A scenario and an ego agent are randomly selected from the dataset for control and CommonRoad-RL [46] is used to get a state description of the ego car and its surroundings. This environment is characterized by a continuous state-action space and stochastic dynamics as the behavior of other agents in the environment is unpredictable and different expert agents act differently based on their preferences. The goal of the agent is to reach its destination at the end of the highway. Figure D3 (Sec. D.3 in the appendix) visualizes the environment. The environment contains two scenarios, one with a minimum distance constraint between the ego car and other vehicles and one with a maximum velocity constraint. The results are depicted in Figure 3. Our method outperforms the other methods by a large margin which is not surprising as the traffic domain is characterized by high stochasticity. The ground truth constraints were arbitrarily chosen by Liu et al. [19] and are in some cases violated in the starting



(a) Point (b) Ant broken

Fig. 4 Comparison of our method against various baselines on the transferability of the cost acquired function. The cost function learned in the "ant" environment is transferred to the "point" agent (left) and the "ant broken" agent (right). The top two plots showcase the obtained rewards, while the bottom two plots depict the constraint violation rates of trajectories sampled from the nominal policy during training. The results are averaged over 10 random seeds, with the x-axis representing the number of timesteps taken in the environment during training. The shaded regions correspond to the standard error.

state of the ego agent. Because of this, no method, even not the expert agent trained given the ground truth constraints, is able to reduce the cost to zero.

5.4 Transfer Learning

When a cost function is learned for a specific agent, it could be beneficial when that cost function can be transferred to other types of agents operating in the same environment. This would prevent us from learning a cost function for every kind of agent. Note that constraints are often environment specific and apply to different types of agents, e.g. traffic rules apply to all vehicles on the road. To this end, we assess the transferability of a learned cost function to other types of agents which possibly have other morphologies and reward functions. We perform the transfer learning experiments proposed by Malik et al. [18]. The cost function learned for the "ant" agent is transferred to a "broken ant" agent for which two of its legs are disabled. We also transfer this cost function to the point agent from OpenAI Gym [44]. The reward function of the point robot encourages the agent to move in counterclockwise in a circle at a distance of 10 units from its initial location. We adopt reward constrained policy optimization [7] to train agents on the given reward function and the transferred cost function. The results are depicted in Figure 4. For both agents our method performs slightly better than ME-ICRL and outperforms VME-ICRL and GACL by a large margin in both reward and number of constraint violations.

Table 1 Ablation study on the pre-training of the feature encoder and the entropy coefficient β, the table contains for every agent the received reward and the constraint violation rate at test time ± standard error, averaged over 10 random seeds and 10 trajectories per seed.

		Ant	Half-cheetah	Swimmer	Walker	Inverted pendulum
Reward	$\beta = 1e{-}5$	5985±259	3895±295	177±52	1647±100	27±3
	$\beta = 1e{-}4$	**7338±211**	**4085±565**	**257±44**	**1858±44**	26±3
	$\beta = 1e{-}3$	6057±313	3499±492	253±54	1746±81	**32±3**
	$\beta = 1e{-}2$	6228±316	2377±631	192±43	1732±77	29±4
	no pre-training	7131±313	3609±490	165±59	1656±71	21±1
Constraint violation rate	$\beta = 1e{-}5$	**0±0**	**0±0**	0.33±0.09	**0±0**	0.22±0.05
	$\beta = 1e{-}4$	**0±0**	**0±0**	**0.002±0.001**	**0±0**	**0.11±0.03**
	$\beta = 1e{-}3$	**0±0**	0.002±0.002	0.12±0.04	**0±0**	.12±0.03
	$\beta = 1e{-}2$	**0±0**	0.024±0.012	0.11±0.05	**0±0**	0.17±0.02
	no pre-training	**0±0**	**0±0**	0.22±0.09	**0±0**	0.23±0.04

5.5 Ablation Studies

5.5.1 Influence of the Hyperparameter β

β determines the weight assigned to the entropy term in the objective function and can be interpreted as a regularization coefficient to enforce randomness into the learned policy π. Higher values of β lead to policies exhibiting reduced bias toward paths with identical cumulative rewards. Consequently, the associated cost function will invalidate a minimal set of state-action pairs required to match feature expectations of the nominal policy with those of the expert. Table 1 reports the reward and the constraint violation rate of trajectories sampled from a policy learned using MCE-ICRL during testing for the virtual robotic environments. We can conclude that the value of β heavily influences the received reward and the number of constraint violations. If β is too small, our method overfits on the expert trajectories leading to small rewards and possibly more constraint violations. For high values of β, the nominal policy becomes more random which also results in lower rewards and more constraint violations. For each environment, the value of β should be tuned to obtain optimal results.

5.5.2 Pre-Training the Feature Encoder

To assess the importance of pre-training the feature encoder, we train a version of MCE-ICRL without pre-training. Table 1 reports the received reward and constraint violation rate of the nominal policy at test time for the virtual robotic environments. From these results it is clear that pre-training the feature encoder provides a significant increase of the received reward. Section F in the appendix provides additional results on the effect of pre-training the feature encoder and β during training and for the other environments.

6 Conclusion and Future Work

We introduced Maximum Causal Entropy Inverse Constrained Reinforcement Learning (MCE-ICRL), a novel ICRL method capable of learning constraints in environments with unknown stochastic dynamics. Our contribution includes a theoretical proof of convergence in a tabular setting, demonstrating the foundational robustness of our approach. Additionally, we presented an approximation allowing our method to effectively scale to complex problems featuring continuous state-action spaces. Our empirical results showcase the superior performance of our method compared to state-of-the-art approaches. Notably, the learned cost functions exhibit successful transferability to diverse agents with distinct reward functions. Despite these achievements, our method does not match expert agent performance in certain scenarios, indicating areas for potential improvement. An ablation study underscores the significance of selecting an appropriately tuned value for the parameter β, suggesting that an automatic tuning mechanism, as proposed by Haarnoja et al. [47], could be a valuable extension to our method. It is crucial to acknowledge the sensitivity of Lagrange relaxation methods to the initialization of Lagrange multipliers and learning rates. Because of this, there is no assurance that the imitation policies can consistently meet the learned constraints. In our current implementation, we considered only hard constraints ($\alpha = 0$), assuming expert demonstrations are devoid of constraint violations. Recognizing that real-world expert agents may not always pursue constraint satisfaction, future research should explore extensions to model soft constraints [17, 25], where expert adherence to constraints is probabilistic. During initial training phases, our agent optimizes reward and entropy without accounting for expert knowledge, leading to inevitable constraint violations. Future investigations should prioritize safe exploration strategies, particularly in safety-critical environments. Our method, does also not guarantee safety when the agent deviates significantly from the optimal path. In such scenarios, uncertainty-aware methods [33, 34] are necessary to ensure safety. We assumed that the expert demonstrations are near optimal. This means that our method might, by mistake, assign higher costs to regions of the state space due to suboptimal demonstrations. To better accommodate these deviations from the expert demonstrations, increasing the budget parameter α could be a promising approach, although, this should still be investigated. Existing benchmarks predominantly focus on simple constraints, often characterized as location constraints. Future work should emphasize the development of benchmarks featuring more intricate constraints, such as those based on the state of multiple objects or constraints defined by specific event sequences. The proposed method can be classified as an online ICRL algorithm as the constrained policy is learned through interaction with the environment. However, realistic applications often only have demonstration data available without a model of the environment. In light of recent advancements in offline Inverse Reinforcement Learning [48–50], extending ICRL to an offline training setting emerges as a critical and promising future direction.

Appendix A Causal Entropy

In this section we elaborate on why it is not advisable to adopt the principle of maximum non-causal entropy (i.e. Shannon entropy) in domains which are characterized by stochastic transition dynamics. Consider the following derivation from the definition of the Shannon entropy of a trajectory:

$$\sum_{t=0}^{T-1} \gamma^t H(s_t, a_t \mid s_{t-1}, a_{t-1}) \tag{A1}$$

$$= \sum_{t=0}^{T-1} \gamma^t H(s_t \mid s_{t-1}, a_{t-1}) + \sum_{t=0}^{T-1} \gamma^t H(a_t \mid s_{t-1}, a_{t-1}) \tag{A2}$$

$$= \sum_{t=0}^{T-1} \gamma^t H(s_t \mid s_{t-1}, a_{t-1}) + \sum_{t=0}^{T-1} \gamma^t H(a_t \mid s_{t-1}). \tag{A3}$$

The non-causal entropy can be written as the sum of the entropy of the transition dynamics (first term of eq. (A3)) and the causal entropy (second term of eq. (A3)). Due to the dependency on transition dynamics, the maximization of Shannon entropy introduces a bias towards actions with uncertain (and potentially hazardous) outcomes. Therefore, in place of conventional Shannon entropy, it is recommended to utilize causal entropy in decision-making processes to mitigate such biases.

Appendix B Minimizing Policy Gradient Variance

The vanilla policy gradient is characterized by minimal bias but tends to exhibit high variance. To mitigate this variance while maintaining low bias, the introduction of a state-dependent baseline is a well-established technique [42].

Proposition 6. *In the gradient of the Lagrangian (eq. (11)), we can replace* $\sum_{t'=t}^{T-1} \gamma^{t'-t}\beta$ *with a state dependent baseline* $b(s_t)$. *Proof. See Section C.5 in the appendix.*

Because of proposition 6 we can replace $\sum_{t'=t}^{T-1} \gamma^{t'-t}\beta$ with the state-value function $V^{\pi_\theta}(s_t)$ which is a commonly used baseline. The gradient can then be written as

$$\nabla_\theta \mathcal{L}(\theta, \lambda) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\big(Q^{\pi_\theta}(s_t, a_t) - \beta \log \pi_\theta(a_t \mid s_t) - V^{\pi_\theta}(s_t)\big)\right] \tag{B4}$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A^{\pi_\theta}(s_t, a_t)\right] \tag{B5}$$

with $A^{\pi_\theta}(s, a)$ the advantage function. We adopt generalized advantage estimation (GAE) [51] to obtain an approximation of the advantage.

Appendix C Proofs

C.1 Proposition 2

The primal objective is defined as

$$\max_{\pi \in \Pi} \mathcal{L}(\pi, \lambda) \tag{C6}$$

with $\mathcal{L}(\pi, \lambda)$ the Lagrangian defined in eq. (4), and thus the optimal policy

$$\pi^* = \arg\max_{\pi \in \Pi} \mathcal{L}(\pi, \lambda). \tag{C7}$$

The simplex Π which restricts the valid policies is defined by two constraints (see eq. (3)). The first constraint requires that $\pi(a \mid s)$ is positive for all states and timesteps. We treat this constraint as implicit since the causal entropy is not defined for negative probabilities. The second constraint requires that π is normalized over all actions for all states and timesteps. We assume, for now, the state-action space is discrete. Augmenting the normalization constraint on the current objective gives us:

$$\begin{aligned} \pi^* &= \arg\max_{\pi \in \Pi} \mathcal{L}(\pi, \lambda) \\ \text{s.t.} \sum_{t=0}^{T-1} \sum_{s_t \in \mathcal{S}} &\left(\sum_{a \in \mathcal{A}} \pi(a \mid s_t) \; - 1 \right) = 0. \end{aligned} \tag{C8}$$

Then, the Lagrangian of the optimization problem in eq. (C8) is defined as

$$\Psi(\pi, \lambda, \mu) = \mathcal{L}(\pi, \lambda) + \sum_{t=0}^{T-1} \sum_{s_t \in \mathcal{S}} \mu_{s_t} \left(\sum_{a \in \mathcal{A}} \pi(a \mid s) \; - 1 \right) \tag{C9}$$

with $\{\mu_{s_t}\}_{s_t \in \mathcal{S}, 0 \leq t \leq T-1}$ the dual variables for the normalization constraint. Then the optimal policy is

$$\pi^* = \arg\max_{\pi} \Psi(\pi, \lambda, \mu). \tag{C10}$$

The optimal policy π^* satisfies the KKT condition:

$$\nabla_\pi \Psi(\pi^*, \lambda, \mu) = 0. \tag{C11}$$

To obtain an expression for the optimal policy, we take the gradient of the Lagrangian w.r.t. the policy, equate $\nabla_\pi \Psi(\pi, \lambda, \mu)$ to zero and solve for π.

Proposition 7. *The gradient of the Lagrangian (eq. (C9)) w.r.t. to the policy $\pi(a_t \mid s_t)$ is given by*

$$\nabla_{\pi(a_t|s_t)} \Psi(\pi, \lambda, \mu) = \mu_{s_t} + \rho_\pi(s_t) \Big(R(s_t, a_t) - \lambda \cdot \phi(s_t, a_t)$$

$$- \beta \log \pi(a_t \mid s_t) - \beta + \mathbb{E}_\pi \Bigg[\sum_{t'=t+1}^{T-1} \gamma^{t'-t} \big(R(S_{t'}, A_{t'}) \\ -\lambda \cdot \phi(S_{t'}, A_{t'}) - \beta \log \pi(A_{t'} \mid S_{t'}) \big) \Big| s_t, a_t \Bigg] \Bigg). \quad \text{(C12)}$$

Proof. We take the gradient of each term of the Lagrangian (eq. (C9)) w.r.t. to the policy $\pi(a_t \mid s_t)$ separately. Uppercase characters S_t, A_t represent random variables while lower case characters s_t, a_t denote individual observations.

$$\nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[R(\tau) \right] \quad \text{(C13)}$$

$$= \nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\sum_{t'=0}^{T-1} \gamma^{t'} R(S_{t'}, A_{t'}) \right] \quad \text{(C14)}$$

$$= \mathbb{E}_{S_t \sim \pi} \left[\nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\sum_{t'=0}^{T-1} \gamma^{t'} R(S_{t'}, A_{t'}) \Big| S_t \right] \right] \quad \text{(C15)}$$

$$= \mathbb{E}_{S_t \sim \pi} \left[\gamma^t \mathbb{I}[S_t = s_t] \nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\sum_{t'=t}^{T-1} \gamma^{t'-t} R(S_{t'}, A_{t'}) \Big| S_t = s_t \right] \right] \quad \text{(C16)}$$

$$= \rho_\pi(s_t) \nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\sum_{t'=t}^{T-1} \gamma^{t'-t} R(S_{t'}, A_{t'}) \Big| S_t = s_t \right] \quad \text{(C17)}$$

$$= \rho_\pi(s_t) \nabla_{\pi(a_t|s_t)} \Bigg(\pi(a_t \mid s_t) \mathbb{E}_\pi \left[\sum_{t'=t}^{T-1} \gamma^{t'-t} R(S_{t'}, A_{t'}) \Big| S_t = s_t, A_t = a_t \right] \\ + \Big(\sum_{a'_t \neq a_t} \pi(a'_t \mid s_t) \Big) \mathbb{E}_\pi \left[\sum_{t'=t}^{T-1} \gamma^{t'-t} R(S_{t'}, A_{t'}) \Big| S_t = s_t, A_t \neq a_t \right] \Bigg) \quad \text{(C18)}$$

$$= \rho_\pi(s_t) \mathbb{E}_\pi \left[\sum_{t'=t}^{T-1} \gamma^{t'-t} R(S_{t'}, A_{t'}) \Big| S_t = s_t, A_t = a_t \right] \quad \text{(C19)}$$

$$= \rho_\pi(s_t) \left[R(s_t, a_t) + \mathbb{E}_\pi \left[\sum_{t'=t+1}^{T-1} \gamma^{t'-t} R(S_{t'}, A_{t'}) \Big| s_t, a_t \right] \right], \quad \text{(C20)}$$

where $\rho_\pi(s_t)$ represents the discounted probability that an agent acting according to policy π will be in state s_t at time t

$$\rho_\pi(s_t) = \mathbb{E}_{S_t \sim \pi} \left[\gamma^t \mathbb{I}[S_t = s_t] \right]. \quad \text{(C21)}$$

The gradient of the entropy term:

$$\nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[H(\tau) \right] \quad \text{(C22)}$$

$$= \nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\sum_{t'=0}^{T-1} \gamma^{t'} H(A_{t'} \mid S_{t'}) \right] \tag{C23}$$

$$= -\nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\sum_{t'=0}^{T-1} \gamma^{t'} \log \pi(A_{t'} \mid S_{t'}) \right] \tag{C24}$$

$$= -\rho_\pi(s_t) \nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\sum_{t'=t}^{T-1} \gamma^{t'-t} \log \pi(A_{t'} \mid S_{t'}) \middle| S_t = s_t \right] \tag{C25}$$

$$= -\rho_\pi(s_t) \nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\log \pi(A_t \mid S_t) + \sum_{t'=t+1}^{T-1} \gamma^{t'-t} \log \pi(A_{t'} \mid S_{t'}) \middle| S_t = s_t \right] \tag{C26}$$

$$= -\rho_\pi(s_t) \left(1 + \log \pi(a_t \mid s_t) + \mathbb{E}_\pi \left[\sum_{t'=t+1}^{T-1} \gamma^{t'-t} \log \pi(A_{t'} \mid S_{t'}) \middle| s_t, a_t \right] \right). \tag{C27}$$

The gradient of the feature matching term:

$$\nabla_{\pi(a_t|s_t)} \lambda \cdot \left(\mathbb{E}_{\mathcal{D}} \left[\sum_{t'=0}^{T-1} \gamma^{t'} \phi(S_t, A_t) \right] - \mathbb{E}_\pi \left[\sum_{t'=0}^{T-1} \gamma^{t'} \phi(S_{t'}, A_{t'}) \right] \right) \tag{C28}$$

$$= -\nabla_{\pi(a_t|s_t)} \lambda \cdot \mathbb{E}_\pi \left[\sum_{t'=0}^{T-1} \gamma^{t'} \phi(S_{t'}, A_{t'}) \right] \tag{C29}$$

$$= -\rho_\pi(s_t) \lambda \cdot \nabla_{\pi(a_t|s_t)} \mathbb{E}_\pi \left[\sum_{t'=0}^{T-1} \gamma^{t'-t} \phi(S_{t'}, A_{t'}) \middle| S_t = s_t \right] \tag{C30}$$

$$= -\rho_\pi(s_t) \lambda \cdot \left[\phi(s_t, a_t) + \mathbb{E}_\pi \left[\sum_{t'=t+1}^{T-1} \gamma^{t'-t} \phi(S_{t'}, A_{t'}) \middle| s_t, a_t \right] \right]. \tag{C31}$$

The gradient of the normalization constraint term:

$$\nabla_{\pi(a_t|s_t)} \sum_{t=0}^{T-1} \sum_{s_t \in \mathcal{S}} \mu_{s_t} \left(\sum_{a \in \mathcal{A}} \pi(a \mid s) \; - 1 \right) \tag{C32}$$

$$= \mu_{s_t}. \tag{C33}$$

After equating the gradient from Proposition 7 to zero and re-arranging, we get an expression for the optimal policy π^*:

$$\pi^*(a_t \mid s_t) = \exp \left(\frac{1}{\beta} \Bigg(R(s_t, a_t) - \lambda \cdot \phi(s_t, a_t) + \frac{\mu_s}{\rho_\pi(s_t)} - \beta \right.$$
$$\left. + \mathbb{E}_\pi \Bigg[\sum_{t'=t+1}^{T-1} \gamma^{t'-t} \big(R(S_{t'}, A_{t'}) - \lambda \cdot \phi(S_{t'}, A_{t'}) \right.$$

$$- \beta \log \pi(A_{t'} \mid S_{t'}))\Big| s_t, a_t\Big]\Big)\Big). \tag{C34}$$

We define $V(s_t)$ and $Q(s_t, a_t)$ as

$$V(s_t) \triangleq \frac{-\mu_{s_t}}{\rho_\pi(s_t)} + \beta \tag{C35}$$

$$Q(s_t, a_t) \triangleq R(s_t, a_t) - \lambda \cdot \phi(s_t, a_t) + \mathbb{E}_\pi\Big[\sum_{t'=t+1}^{T-1} \gamma^{t'-t}\big(R(S_{t'}, A_{t'}) - \lambda \cdot \phi(S_{t'}, A_{t'}) - \beta \log \pi(A_{t'} \mid S_{t'})\Big| s_t, a_t\Big]. \tag{C36}$$

The optimal policy can then be described as

$$\pi^*(a_t \mid s_t) = \exp\Big(\frac{1}{\beta}\big(Q(s_t, a_t) - V(s_t)\big)\Big). \tag{C37}$$

In eq. (3) we defined the set of valid policies as the set of normalized policies Π and assume all values of μ are chosen such that $\sum_{a\in\mathcal{A}} \pi(a \mid s) = 1 \ \forall s \in \mathcal{S}$. Because of this, we can say that $\forall s \in \mathcal{S}$:

$$1 = \sum_{a\in\mathcal{A}} \pi(a \mid s) \tag{C38}$$

$$= \sum_{a\in\mathcal{A}} \exp\Big(\frac{1}{\beta}(Q(s, a) - V(s))\Big) \tag{C39}$$

$$\exp(\frac{1}{\beta}V(s)) = \sum_{a\in\mathcal{A}} \exp\Big(\frac{1}{\beta}(Q(s, a) - V(s))\Big) \exp(\frac{1}{\beta}V(s)) \tag{C40}$$

$$= \sum_{a\in\mathcal{A}} \exp\Big(\frac{1}{\beta}Q(s, a)\Big) \tag{C41}$$

$$V(s) = \beta \log \sum_{a\in\mathcal{A}} \exp\Big(\frac{1}{\beta}Q(s, a)\Big). \tag{C42}$$

For a continuous action space we get

$$V(s) = \beta \log \int \exp\Big(\frac{1}{\beta}Q(s, a)\Big)\, \mathrm{d}a. \tag{C43}$$

Similarly, it can be demonstrated that

$$Q(s_t, a_t) = R(s_t, a_t) - \lambda \cdot \phi(s_t, a_t)$$

$$+\mathbb{E}_\pi\left[\sum_{t'=t+1}^{T-1}\gamma^{t'-t}\big(R(S_{t'},A_{t'})-\lambda\cdot\phi(S_{t'},A_{t'})-\beta\log\pi(A_{t'}\mid S_{t'})\big)\middle|s_t,a_t\right] \tag{C44}$$

$$=R(s_t,a_t)-\lambda\cdot\phi(s_t,a_t)$$
$$+\gamma\,\mathbb{E}_p\left[\mathbb{E}_\pi\Big[Q(S_{t+1},A_{t+1})-\beta\log\pi(A_{t+1}\mid S_{t+1})\Big]\middle|s_t,a_t\right] \tag{C45}$$

$$=R(s_t,a_t)-\lambda\cdot\phi(s_t,a_t)$$
$$+\gamma\,\mathbb{E}_p\left[\mathbb{E}_\pi\Big[Q(S_{t+1},A_{t+1})-(Q(S_{t+1},A_{t+1})-V(S_{t+1}))\Big]\middle|s_t,a_t\right] \tag{C46}$$

$$=R(s_t,a_t)-\lambda\cdot\phi(s_t,a_t)+\gamma\,\mathbb{E}_p\Big[V(S_{t+1})\Big|s_t,a_t\Big]. \tag{C47}$$

C.2 Theorem 3

First, we will prove the convergence of the policy evaluation step.

Proposition 8 (Policy Evaluation)**.** *Starting with an initial Q-function $Q^0:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ with $|\mathcal{A}|<\infty$, where $Q^{k+1}=\mathcal{T}^\pi Q^k$. Then the sequence Q^k will converge to the Q-value of π as $k\to\infty$.*

Proof.
We define an augmented reward signal as

$$R_\pi(s_t,a_t)\triangleq R(s_t,a_t)-\lambda\cdot\phi(s_t,a_t)+\beta H(a_t\mid s_t). \tag{C48}$$

Applying the standard convergence results for policy evaluation [52] proves the proposition. It is necessary to have $|\mathcal{A}|<\infty$ in order to ensure that the augmented reward is bounded.

For the projection presented in eq. (10), we can prove that the new, projected policy has a higher value than the old policy according to the objective in eq. (4).
Proposition 9 (Policy Improvement)**.** *Given $\pi_{old}\in\Pi$ and π_{new} be the policy obtained by solving the minimization problem defined in eq. (10). Then $Q^{\pi_{new}}(s_t,a_t)\geq Q^{\pi_{old}}(s_t,a_t)$ for all $(s_t,a_t)\in\mathcal{S}\times\mathcal{A}$ with $|\mathcal{A}|<\infty$.*

Proof.
Let $Q^{\pi_{old}}$ and $V^{\pi_{old}}$ be the action-value and state-value function corresponding to $\pi_{old}\in\Pi$. We can define $J_{\pi_{old}}$ as

$$J_{\pi_{old}}\big(\pi'(\cdot\mid s_t)\big)=D_{KL}\left(\pi'\;||\;\exp\Big(\frac{1}{\beta}\left(Q^{\pi_{old}}(s_t\mid\cdot)-V^{\pi_{old}}(s_t)\right)\Big)\right). \tag{C49}$$

Then π_{new} can be defined as

$$\pi_{new}(\cdot\mid s_t)=\arg\min_{\pi'\in\Pi}J_{\pi_{old}}\big(\pi'(\cdot\mid s_t)\big). \tag{C50}$$

The inequality $J_{\pi_{old}}\big(\pi_{new}(\cdot \mid s_t)\big) \leq J_{\pi_{old}}\big(\pi_{old}(\cdot \mid s_t)\big)$ *must hold, as we have the option of always selecting* $\pi_{new} = \pi_{old}$ *from the set of policies* Π*. Hence,*

$$\mathbb{E}_{\pi_{new}}\left[\log \pi_{new}(a_t \mid s_t) - \frac{1}{\beta}(Q^{\pi_{old}}(s_t, a_t) - V^{\pi_{old}}(s_t))\right] \leq$$
$$\mathbb{E}_{\pi_{old}}\left[\log \pi_{old}(a_t \mid s_t) - \frac{1}{\beta}(Q^{\pi_{old}}(s_t, a_t) - V^{\pi_{old}}(s_t))\right] \tag{C51}$$

$$\mathbb{E}_{\pi_{new}}\left[\log \pi_{new}(a_t \mid s_t) - \frac{1}{\beta}Q^{\pi_{old}}(s_t, a_t)\right] + \frac{1}{\beta}V^{\pi_{old}}(s_t) \leq$$
$$\mathbb{E}_{\pi_{old}}\left[\log \pi_{old}(a_t \mid s_t) - \frac{1}{\beta}Q^{\pi_{old}}(s_t, a_t)\right] + \frac{1}{\beta}V^{\pi_{old}}(s_t) \tag{C52}$$

$$\mathbb{E}_{\pi_{new}}\left[\log \pi_{new}(a_t \mid s_t) - \frac{1}{\beta}Q^{\pi_{old}}(s_t, a_t)\right] \leq$$
$$\mathbb{E}_{\pi_{old}}\left[\log \pi_{old}(a_t \mid s_t) - \frac{1}{\beta}Q^{\pi_{old}}(s_t, a_t)\right] \tag{C53}$$

$$\mathbb{E}_{\pi_{new}}\left[Q^{\pi_{old}}(s_t, a_t) - \beta \log \pi_{new}(a_t \mid s_t)\right] \geq V^{\pi_{old}}(s_t). \tag{C54}$$

We can bring $V^{\pi_{old}}$ *outside the expectation since the state-value function only depends on the current state (eq. (C52)). Eq. (C54) follows from the definition of the optimal policy in eq. (6). We can repeatedly expand the Bellman equation by applying the Bellman equation and the inequality of eq. (C54):*

$$\begin{aligned}
&Q^{\pi_{old}}(s_t, a_t) \\
&= R(s_t, a_t) - \lambda \cdot \phi(s_t, a_t) + \gamma\, \mathbb{E}_p\left[V^{\pi_{old}}(s_{t+1})\right] && \text{(C55)} \\
&\leq R(s_t, a_t) - \lambda \cdot \phi(s_t, a_t) + \gamma\, \mathbb{E}_{p,\pi_{new}}\big[\\
&\qquad Q^{\pi_{old}}(s_{t+1}, a_{t+1}) - \beta \log \pi_{new}(a_{t+1} \mid s_{t+1})\big] && \text{(C56)} \\
&\vdots \\
&\leq Q^{\pi_{new}}(s_t, a_t). && \text{(C57)}
\end{aligned}$$

Then, the convergence to $Q^{\pi_{new}}$ *follows from Proposition 8.*

At each iteration i, the policy π_i will reach an optimum due to Proposition 9. Because the sequence Q^{π_i} increases monotonically and is bounded for $\pi \in \Pi$, the sequence converges to some π^*. We have to prove that π^* is indeed optimal. At convergence, it must be the case that $J_{\pi^*}(\pi^*(\cdot \mid s_t)) < J_{\pi^*}(\pi(\cdot \mid s_t))$ for all $\pi \in \Pi$, $\pi \neq \pi^*$. By repeatedly expanding Q^{π^*} as done in Proposition 9, we can show that $Q^{\pi^*}(s_t, a_t) > Q^{\pi}(s_t, a_t)$ for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. Thus, the value of any other policy in Π is lower than that of the converged policy. This makes π^* optimal in Π.

C.3 Proposition 5

$\max_{\pi \in \Pi} \mathcal{L}(\pi, \lambda)$ can be viewed as a pointwise maximum of affine functions of λ, thus is concave. $\lambda \geq 0$ is an affine constraint. Hence, the dual problem (eq. (5)) is a convex optimization problem.

C.4 Proposition 4

We calculate the gradient of each term of the Lagrangian eq. (4) separately w.r.t. the policy parameters θ. The gradient of the causal entropy term is

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}\big[H(\tau)\big] \tag{C58}$$

$$= \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t H(a_t \mid s_t)\right] \tag{C59}$$

$$= -\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t \log \pi_\theta(a_t \mid s_t)\right] \tag{C60}$$

$$= -\nabla_\theta \sum_\tau \left(P(\tau \mid \pi_\theta) \sum_{t=0}^{T-1} \gamma^t \log \pi_\theta(a_t \mid s_t)\right) \tag{C61}$$

$$\begin{aligned} = -\sum_\tau \Bigg(&\nabla_\theta P(\tau \mid \pi_\theta) \sum_{t=0}^{T-1} \gamma^t \log \pi_\theta(a_t \mid s_t) \\ &+ P(\tau \mid \pi_\theta) \sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Bigg) \end{aligned} \tag{C62}$$

$$\begin{aligned} = -\sum_\tau \Bigg(&P(\tau \mid \pi_\theta) \nabla_\theta \log P(\tau \mid \pi_\theta) \sum_{t=0}^{T-1} \gamma^t \log \pi_\theta(a_t \mid s_t) \\ &+ P(\tau \mid \pi_\theta) \sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Bigg) \end{aligned} \tag{C63}$$

$$\begin{aligned} = -\sum_\tau P(\tau \mid \pi_\theta) \Bigg(&\nabla_\theta \log P(\tau \mid \pi_\theta) \sum_{t=0}^{T-1} \gamma^t \log \pi_\theta(a_t \mid s_t) \\ &+ \sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Bigg) \end{aligned} \tag{C64}$$

$$= -\mathbb{E}_{\tau \sim \pi_\theta}\left[\nabla_\theta \log P(\tau \mid \pi_\theta) \sum_{t=0}^{T-1} \gamma^t \log \pi_\theta(a_t \mid s_t) + \sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \tag{C65}$$

$$= -\mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t'=t}^{T-1} \gamma^{t'-t} \log \pi_\theta(a_{t'} \mid s_{t'})\right.$$

$$+\sum_{t=0}^{T-1}\gamma^t\nabla_\theta\log\pi_\theta(a_t\mid s_t)\Big] \tag{C66}$$

$$=-\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\Big(\sum_{t'=t}^{T-1}\gamma^{t'-t}\log\pi_\theta(a_{t'}\mid s_{t'})+1\Big)\Big]. \tag{C67}$$

In eq. (C59) and eq. (C60), we apply the definition of causal entropy. $P(\tau\mid\pi)$ denotes the probability of a trajectory τ under policy π. Eq. (C63) follows from $\nabla P(\tau\mid\pi)=P(\tau\mid\pi)\nabla\log P(\tau\mid\pi)$. Eq. (C66) follows from the definition of the probability of a trajectory τ under a policy π: $P(\tau\mid\pi)=\mathcal{I}(s_0)\prod_{t=0}^{T-1}p(s_{t+1}\mid s_t,a_t)\pi(a_t\mid s_t)$. Then the second sum needs to be updated since the policy at time t' cannot affect the policy at time t when $t<t'$ because of the causal relation between consecutive states. The gradient of the reward term can be calculated similarly

$$\nabla_\theta\mathbb{E}_{\tau\sim\pi_\theta}\big[R(\tau)\big] \tag{C68}$$

$$=\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi(a_t\mid s_t)\sum_{t'=t}^{T-1}\gamma^{t'-t}R(s_t,a_t)\Big], \tag{C69}$$

and the feature matching term:

$$\nabla_\theta\lambda\cdot\Big(\mathbb{E}_{\tau\sim\mathcal{D}}\Big[\sum_{t=0}^{T-1}\gamma^t\phi(s_t,a_t)\Big]-\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\gamma^t\phi(s_t,a_t)\Big]-\boldsymbol{\alpha}_k\Big) \tag{C70}$$

$$=-\nabla_\theta\lambda\cdot\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\gamma^t\phi(s_t,a_t)\Big] \tag{C71}$$

$$=-\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi(a_t\mid s_t)\sum_{t'=t}^{T-1}\gamma^{t'-t}\lambda\cdot\phi(s_t,a_t)\Big]. \tag{C72}$$

In our work, the same discount factor is used to discount rewards, costs and entropies. Putting it all together, we get

$$\nabla_\theta\mathcal{L}(\theta,\lambda)=\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\sum_{t'=t}^{T-1}\gamma^{t'-t}\big(R(s_{t'},a_{t'})-\lambda\cdot\phi(s_{t'},a_{t'})-\beta\log\pi_\theta(a_{t'}\mid s_{t'})-\beta\big)\Big]. \tag{C73}$$

We rewrite eq. (C73)

$$\nabla_\theta\mathcal{L}(\theta,\lambda)$$
$$=\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\Big(R(s_t,a_t)-\lambda\cdot\phi(s_t,a_t)$$

$$+\sum_{t'=t+1}^{T-1}\gamma^{t'-t}\big(R(s_{t'},a_{t'})-\lambda\cdot\phi(s_{t'},a_{t'})-\beta\log\pi_\theta(a_{t'}\mid s_{t'})\big)$$
$$-\beta\log\pi_\theta(a_t\mid s_t)-\sum_{t'=t}^{T-1}\gamma^{t'-t}\beta\Big)\Big]. \quad \text{(C74)}$$

Using the definition of the action-value function (eq. (7)) we obtain:

$$\nabla_\theta\mathcal{L}(\theta,\lambda)=\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\big(Q(s_t,a_t)-\beta\log\pi_\theta(a_t\mid s_t)-\sum_{t'=t}^{T-1}\gamma^{t'-t}\beta\big)\Big]. \quad \text{(C75)}$$

C.5 Proposition 6

We can replace $\sum_{t'=t}^{T-1}\gamma^{t'-t}\beta$ with a state dependent baseline $b(s_t)$ in eq. (11) only when this does not affect the gradient $\nabla_\theta\mathcal{L}(\theta,\lambda)$. We rewrite eq. (11):

$$\nabla_\theta\mathcal{L}(\theta,\lambda) \quad \text{(C76)}$$
$$=\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\big(Q(s_t,a_t)-\beta\log\pi_\theta(a_t\mid s_t)-\sum_{t'=t}^{T-1}\gamma^{t'-t}\beta\big)\Big] \quad \text{(C77)}$$
$$=\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\big(Q(s_t,a_t)-\beta\log\pi_\theta(a_t\mid s_t)\big)\Big]$$
$$-\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi(a_t\mid s_t)\sum_{t'=t}^{T-1}\gamma^{t'-t}\beta\Big] \quad \text{(C78)}$$

In the second term, we replace $\sum_{t'=t}^{T-1}\gamma^{t'-t}\beta$ with a state-dependent baseline $b(s_t)$ which gives us:

$$\nabla_\theta\mathcal{L}(\theta,\lambda)=\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi_\theta(a_t\mid s_t)\big(Q(s_t,a_t)-\beta\log\pi_\theta(a_t\mid s_t)\big)\Big]$$
$$-\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi(a_t\mid s_t)b(s_t)\Big] \quad \text{(C79)}$$

Since the introduced baseline only affects the second term it suffices to prove that

$$\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi(a_t\mid s_t)\sum_{t'=t}^{T-1}\gamma^{t'-t}\beta\Big]=\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta\log\pi(a_t\mid s_t)b(s_t)\Big] \quad \text{(C80)}$$

$$\mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t \mid s_t) b(s_t)\Big] \tag{C81}$$

$$= \mathbb{E}_{s_{0:T-1},a_{0:T-1}}\Big[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t \mid s_t) b(s_t)\Big] \tag{C82}$$

$$= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[\sum_{t=0}^{T-1}\mathbb{E}_{s_{t+1:T-1},a_{t:T-1}}\big[\nabla_\theta \log \pi_\theta(a_t \mid s_t) b(s_t)\big]\Big] \tag{C83}$$

$$= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[\sum_{t=0}^{T-1} b(s_t)\,\mathbb{E}_{s_{t+1:T-1},a_{t:T-1}}\big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\big]\Big] \tag{C84}$$

$$= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[\sum_{t=0}^{T-1} b(s_t)\,\mathbb{E}_{a_t}\big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\big]\Big] \tag{C85}$$

$$= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[\sum_{t=0}^{T-1} b(s_t)\sum_{a_t\in\mathcal{A}}\pi_\theta(a_t \mid s_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big]\Big] \tag{C86}$$

$$= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[\sum_{t=0}^{T-1} b(s_t)\sum_{a_t\in\mathcal{A}}\nabla_\theta \pi_\theta(a_t \mid s_t)\Big]\Big] \tag{C87}$$

$$= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[\sum_{t=0}^{T-1} b(s_t)\nabla_\theta \sum_{a_t\in\mathcal{A}}\pi_\theta(a_t \mid s_t)\Big]\Big] \tag{C88}$$

$$= \mathbb{E}_{s_{0:t},a_{0:t-1}}\Big[\sum_{t=0}^{T-1} b(s_t)\nabla_\theta \cdot 1\Big]\Big] \tag{C89}$$

$$= 0 \tag{C90}$$

$\mathbb{E}_{\tau\sim\pi_\theta}\big[\sum_{t=0}^{T-1}\nabla_\theta \log \pi(a_t \mid s_t)\sum_{t'=t}^{T-1}\gamma^{t'-t}\beta\big]$ can be solved similarly and also results in 0.

Appendix D Details on the Environments

In this section we provide details on and screenshots from the environments used for evaluation.

D.1 Gridworld

Figure D1 shows the gridworld used in the experiments, with the yellow "I" representing the initial state, the yellow "O" the goal state and the red crosses the constrained states.

D.2 Virtual Robotics Environment

Figure D.3 shows screenshots of the realistic robotic environments.

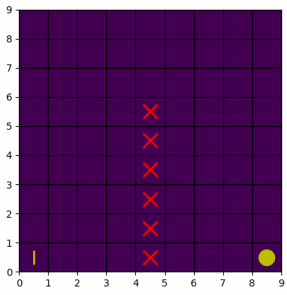
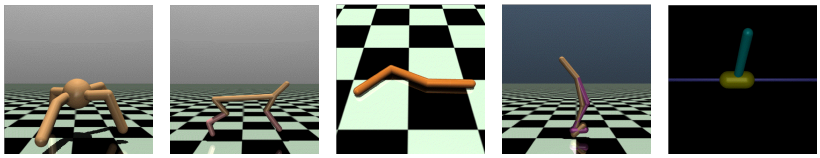


Fig. D1 Gridworld environment



(a) Ant (b) Half-cheetah (c) Swimmer (d) Walker (e) Inverted pendulum

Fig. D2 Screenshots from the virtual robotics environments.

Fig. D3 Realistic traffic environment

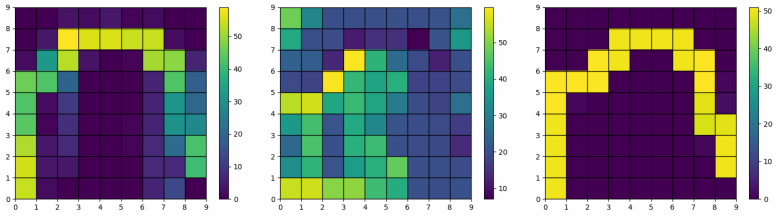
D.3 Realistic Traffic Environment

Figure D3 shows the realistic traffic environment (image from original publication [19]). The ego car is shown in blue, other cars are red. The agent only has partial state information, observations are restricted by an observation radius (visualized in blue around the ego agent). The agent's destination is shown in yellow.

Appendix E Experimental Settings

All experiments were run on a NVIDIA GeForce GTX 980 GPU with 3 GB RAM. We adopted Adam [53] to optimize all neural networks. To calculate the nominal policy, we use Proximal Policy Optimization (PPO) [43]. The output layer of the policy network utilizes a soft-max activation function such that the requirements of the policy specified in eq. 3 are met.

The ϕ_ζ network is a standard feedforward neural network with ReLU activation on the hidden layers and Sigmoid activation on the output layer. λ and ζ are adjusted using possibly different learning rates. Every iteration the learning rate for ζ is updated using an exponential decay schedule parameterized by a decay factor. Table E1 reports an



(a) Expert trajectories state visitations (b) Recovered cost function (c) State visitations for trajectories sampled from the nominal policy

Fig. F4 Results from MCE-ICRL ($\beta = 0.01$) in the proposed gridworld environment.

overview of the used hyperparameters for our method. Table E2, E3 and E4 provide the used hyperparameters for the baseline methods GACL, ME-ICRL and VME-ICRL respectively.

Appendix F Additional Experimental Results

F.1 Gridworld

Figure 4(a) shows the states visited by the expert which are used as input for our method. Figure 4(b) reports the learned cost function when using MCE-ICRL with $\beta = 0.01$. Figure 4(c) shows the state visitations of the nominal agent. Figure F7 depicts the reward and constraint violation rate during training at different timesteps for the different methods for different rates of stochasticity. Figure F8 depicts the reward and constraint violation rate during training at different timesteps for our method with various value of β and for different rates of stochasticity. Figure F9 depicts the reward received and the constraint violation rate during training for the ablation study on the pre-training of the feature encoder.

F.2 Virtual Robotics Environments

Figure F10 depicts the reward received and the constraint violation rate during training for different values of β. Figure F11 depicts the reward received and the constraint violation rate during training for the ablation study on the pre-training of the feature encoder.

F.3 Realistic Traffic Environment

Figure F5 depicts the reward and constraint violation rate during training at different timesteps for our method with various value of β. Figure F6 depicts the reward received and the constraint violation rate during training for the ablation study on the pre-training of the feature encoder.

Table E1 Overview of the used hyperparameters for MCE-ICRL (our method)

	Ant	Cheetah	Swimmer	Walker	Pendulum	Gridworld	HighD dist.	HighD vel.
General								
α	0	0	0	0	0	0	0	0
β	0.0001	0.0001	0.0001	0.0001	0.0001	0.00001	0.0001	0.0001
γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Iterations	20	30	100	50	200	20	100	50
Expert trajectories	20	10	50	50	50	50	20	50
Max. trajectory length	500	1000	500	500	100	200	1000	1000
PPO								
Batch size	128	128	128	64	64	64	64	64
Learning rate	3.00E-05	3.00E-05	3.00E-05	0.0003	0.0001	3.00E-05	0.0001	0.0001
Steps	2048	2048	2048	2048	2048	2048	1024	1024
Epochs	20	20	20	10	20	10	20	20
Timesteps	200000	200000	50000	200000	100000	50000	50000	25000
GAE-λ	0.9	0.9	0.9	0.95	0.9	0.9	0.9	0.9
Target KL	0.02	0.02	0.02	0.01	0.01	0.01	0.02	0.02
Bootstrap timesteps	200000	200000	200000	200000	100000	35000	20000	0
Policy network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
λ, ζ								
Learning rate decay factor ζ	0.9	0.9	0.9	1	0.9	1	0.9	0.9
Batch size	64	64	64	64	128	64	1000	1000
Network ϕ_ζ	[40]	[40]	[40]	[40]	[64]	[40, 40]	[20]	[40, 20, 10]
Feature dimensions	64	64	64	64	16	64	10	8
Learning rate ζ	0.0003	0.0003	0.0003	0.0003	0.0005	0.0005	0.0001	1.00E-06
Learning rate λ	0.0001	0.0001	0.0001	0.0001	0.0001	0.0005	0.0001	1.00E-06
Bootstrap iterations	20	200	200	200	5	20	20	0
λ initial values	1	1	1	1	1	1	1	1

Table E2 Overview of the used hyperparameters for GACL

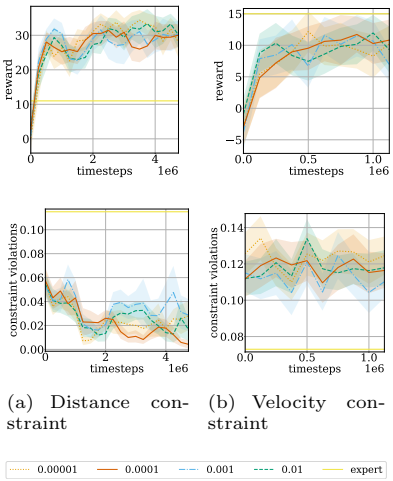
	Ant	Cheetah	Swimmer	Walker	Pendulum	Gridworld	HighD dist.	HighD vel.
Policy								
Batch size	128	64	64	64	128	64	64	64
Target KL	0.02	0.01	0.01	0.01	0.02	0.01	0.01	0.01
Learning rate	3.00E-05	0.0003	0.0003	0.0003	3.00E-05	0.0003	0.0005	0.0003
GAE-γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE-λ	0.9	0.95	0.95	0.95	0.9	0.95	0.95	0.95
Policy network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
Value network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
Discriminator								
Architecture	[40, 40]	[30]	[30]	[64, 64]	[40, 40]	[40,40]	[30]	[30]
Learning rate	0.005	0.003	0.003	0.003	0.00001	0.003	0.003	0.003

Table E3 Overview of the used hyperparameters for ME-ICRL

	Ant	Cheetah	Swimmer	Walker	Pendulum	Gridworld	HighD dist.	HighD vel.
Policy								
Batch size	128	64	128	128	128	64	64	64
Target KL	0.02	0.01	0.02	0.02	0.02	0.01	0.01	0.01
Learning rate	3.00E-05	3.00E-05	3.00E-05	0.0001	3.00E-05	0.0003	0.0001	0.0001
Timesteps	200000	200000	50000	200000	100000	50000	5000	5000
Reward-GAE-γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Reward-GAE-λ	0.9	0.95	0.9	0.9	0.9	0.95	0.95	0.95
Cost-GAE-γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Cost-GAE-λ	0.9	0.95	0.9	0.9	0.9	0.95	0.95	0.95
Policy network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
Value network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
Cost value network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
Lagrangian λ								
Initial values	0.1	1	0.1	0.1	0.1	1	1	1
Learning rate	0.05	0.1	0.05	0.05	0.01	0.1	0.01	0.01
Budget λ	0	0	0	0	0	0	0	0
Constraint function,								
Architecture	[40, 40]	[20]	[40,40]	[64, 64]	[20]	[20,20]	[20]	[20]
Learning rate	0.005	0.05	0.005	0.001	0.001	0.05	0.0005	0.005
Backward iterations	5	10	5	5	5	10	10	10
Regularizer weight	0.6	0.5	0.6	0.6	0.6	0.5	0.5	0.5
Max forward KL	10	10	10	10	10	10	10	10
Max backward KL	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5

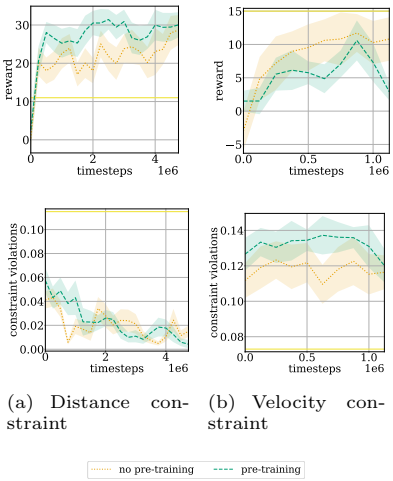
Table E4 Overview of the used hyperparameters for VME-ICRL

	Ant	Cheetah	Swimmer	Walker	Pendulum	Gridworld	HighD dist.	HighD vel.
β-distribution								
α	0.9	0.9	0.9	0.009	0.09	0.9	0.9	0.9
β	0.1	0.1	0.1	0.001	0.01	0.1	0.1	0.1
Policy								
Batch size	128	64	128	128	128	64	64	64
Target KL	0.02	0.01	0.01	0.02	0.02	0.01	0.01	0.01
Learning rate	3.00E-05	3.00E-05	0.0003	0.0001	3.00E-05	0.0003	0.0001	0.0001
Timesteps	200000	200000	50000	200000	100000	50000	5000	5000
Reward-GAE-γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Reward-GAE-λ	0.9	0.95	0.9	0.9	0.9	0.95	0.95	0.95
Cost-GAE-γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Cost-GAE-λ	0.9	0.95	0.95	0.9	0.9	0.95	0.95	0.95
Policy network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
Value network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
Cost value network	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64, 64]	[64,64]	[64, 64]	[64, 64]
Lagrangian λ								
Initial values	0.1	1	5	0.1	0.1	1	1	1
Learning rate	0.05	0.1	0.01	0.05	0.01	0.1	0.01	0.01
Budget λ	0	0	0	0	0	0	0	0
Constraint function,								
Architecture	[40, 40]	[20]	[20]	[64, 64]	[20]	[20,20]	[20]	[20]
Learning rate	0.005	0.005	0.001	0.005	0.0001	0.05	0.0005	0.005
Backward iterations	5	10	5	5	5	10	10	10
Regularizer weight	0.6	0.5	0.6	0.6	0.6	0.5	0.5	0.5
Max forward KL	10	10	10	10	10	10	10	10
Max backward KL	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5



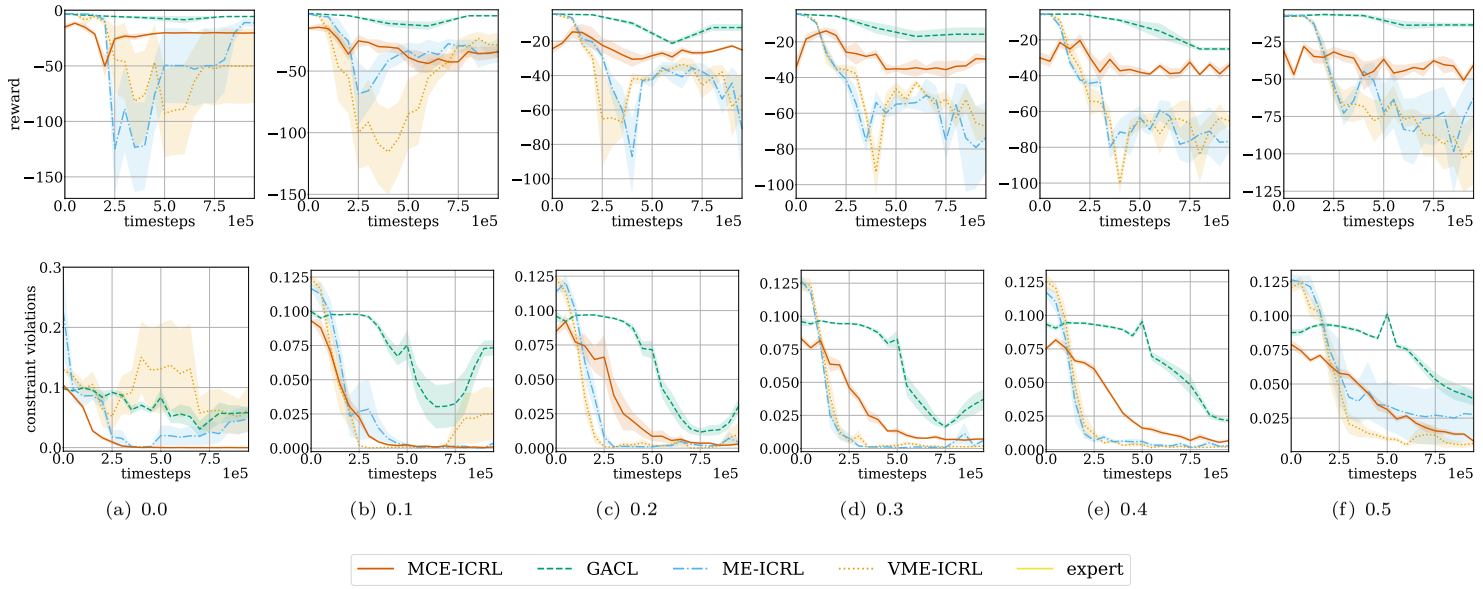
(a) Distance constraint (b) Velocity constraint

Fig. F5 Evaluation of our method in the realistic traffic environment for different values of β: reward (top) and constraint violation rate (bottom) of trajectories sampled from the nominal policy during training. Results are averaged over 10 random seeds. The x-axis is the number of timesteps taken in the environment during training. The shaded regions correspond with the standard error.



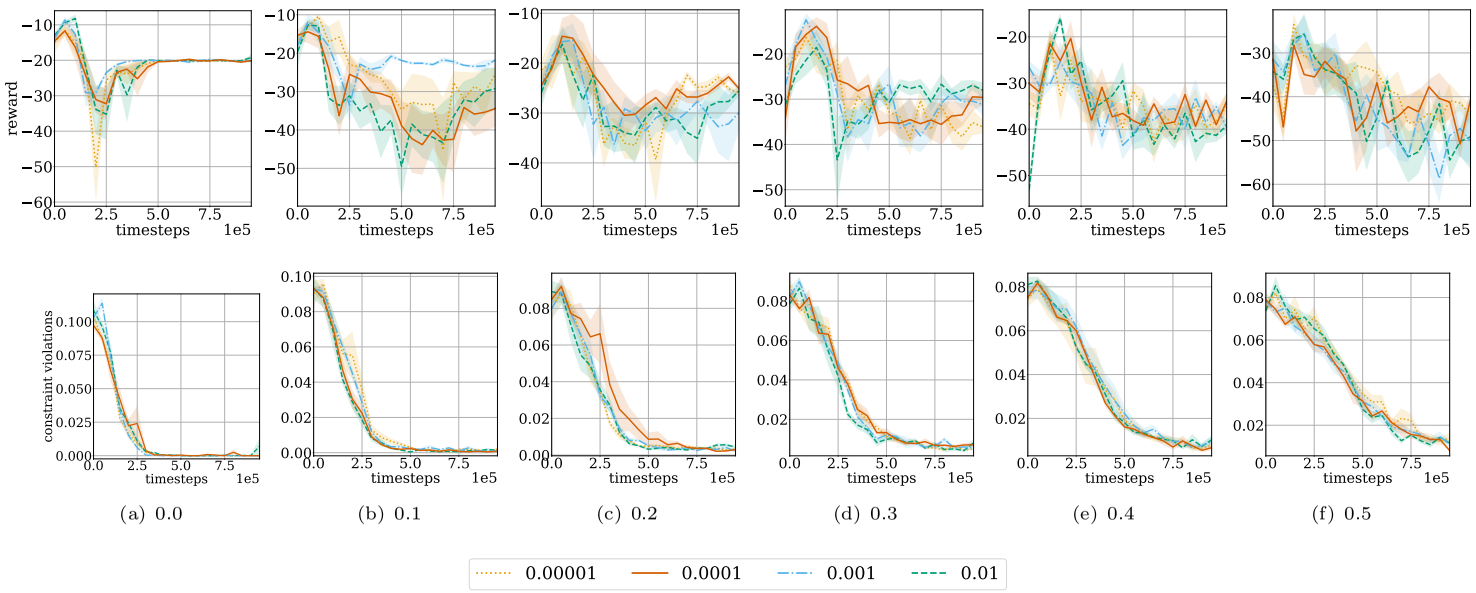
(a) Distance constraint (b) Velocity constraint

Fig. F6 Evaluation of our method in the realistic traffic environment for feature encoder pre-training disabled: reward and constraint violation rate of trajectories sampled from the nominal policy during training. Results are averaged over 10 random seeds. The x-axis is the number of timesteps taken in the environment during training. The shaded regions correspond with the standard error.



(a) 0.0 (b) 0.1 (c) 0.2 (d) 0.3 (e) 0.4 (f) 0.5

Fig. F7 Evaluation of the different methods in the gridword environment for different rates of stochasticity $(0.0, 0.1, 0.2, 0.3, 0.4, 0.5)$: reward (top) and constraint violation rate (bottom) of trajectories sampled from the nominal policy during training. Results are averaged over 5 random seeds. The x-axis is the number of timesteps taken in the environment during training. The shaded regions correspond with the standard error.



(a) 0.0 (b) 0.1 (c) 0.2 (d) 0.3 (e) 0.4 (f) 0.5

Fig. F8 Evaluation of the different methods in the gridword environment for different rates of stochasticity $(0.0, 0.1, 0.2, 0.3, 0.4, 0.5)$ and different values of β $(0.00001, 0.0001, 0.001, 0.01)$: reward and constraint violation rate of trajectories sampled from the nominal policy during training. Results are averaged over 5 random seeds. The x-axis is the number of timesteps taken in the environment during training. The shaded regions correspond with the standard error.

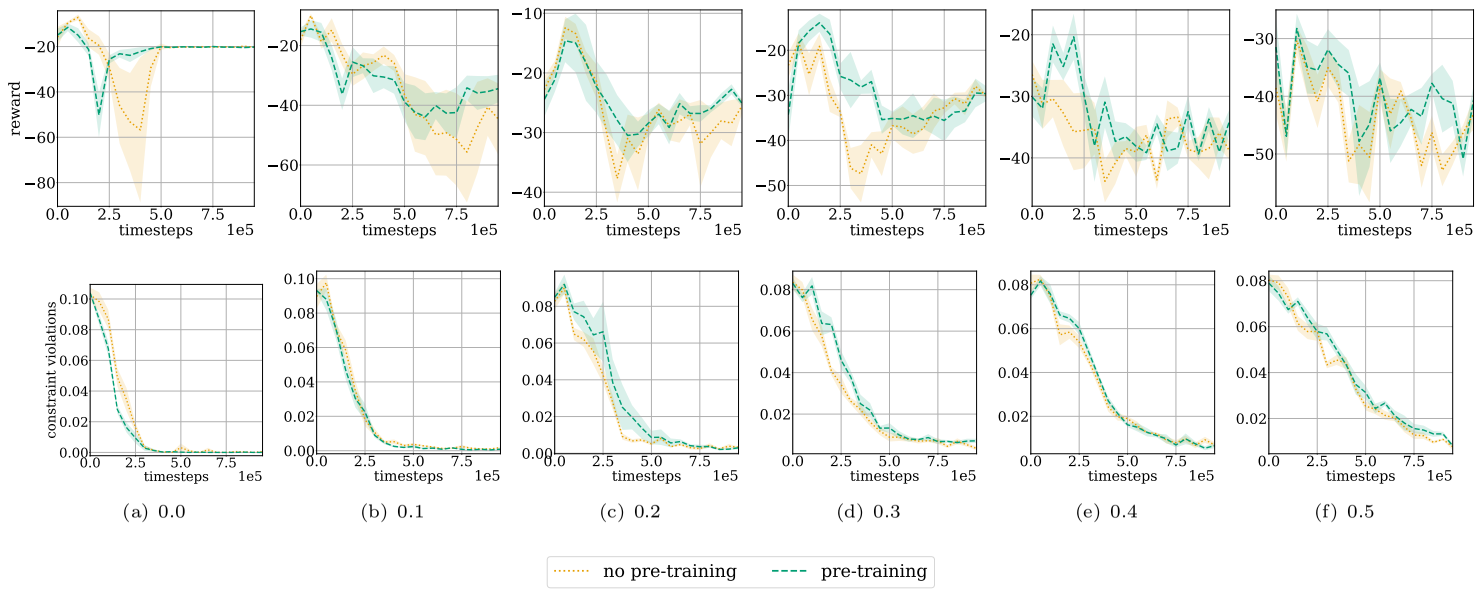
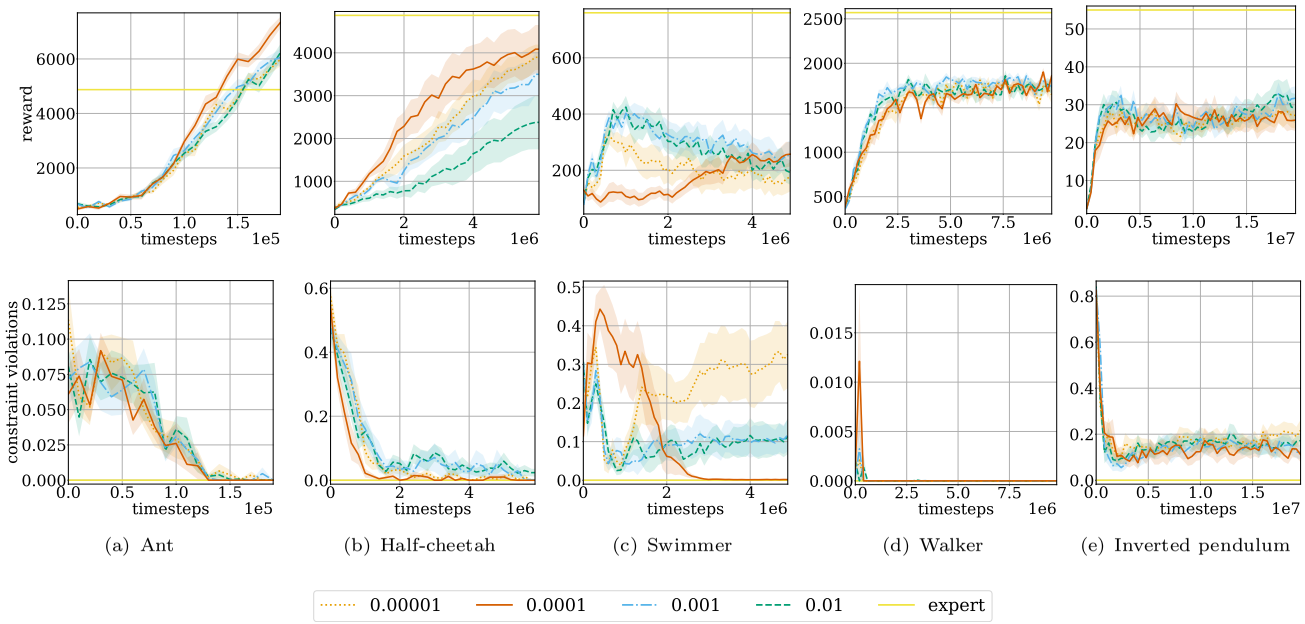
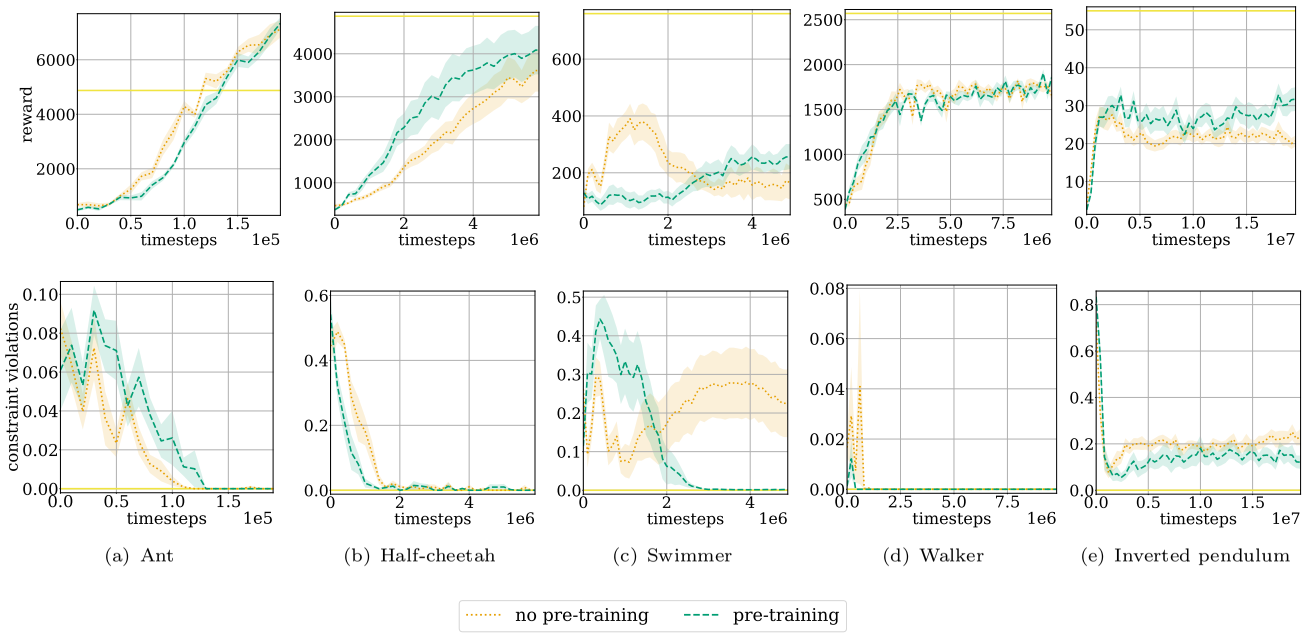


Fig. F9 Ablation study on feature encoder pre-training in the gridworld environment for different rates of stochasticity $(0.0, 0.1, 0.2, 0.3, 0.4, 0.5)$: reward and constraint violation rate of trajectories sampled from the nominal policy during training. Results are averaged over 5 random seeds. The x-axis is the number of timesteps taken in the environment during training. The shaded regions correspond with the standard error.



(a) Ant (b) Half-cheetah (c) Swimmer (d) Walker (e) Inverted pendulum

Fig. F10 Evaluation of our method in the virtual robotics environments for different values of β: reward and constraint violation rate of trajectories sampled from the nominal policy during training. Results are averaged over 10 random seeds. The x-axis is the number of timesteps taken in the environment during training. The shaded regions correspond with the standard error.



(a) Ant (b) Half-cheetah (c) Swimmer (d) Walker (e) Inverted pendulum

Fig. F11 Ablation study on feature encoder pre-training in the virtual robotics environments: reward and constraint violation rate of trajectories sampled from the nominal policy during training. Results are averaged over 10 random seeds. The x-axis is the number of timesteps taken in the environment during training. The shaded regions correspond with the standard error.

Statements and Declarations

Funding

This research was partially funded by the Flemish Government (Flanders AI Research Program).

Conflicts of interest/Competing interests

The authors have no relevant financial or non-financial interests to disclose.

Ethics approval and consent to participate

Not Applicable.

Consent for publication

Not Applicable.

Data availability

All expert trajectories are available at https://gitlab.ilabt.imec.be/mwbaert/maxcausalent-icrl

Materials availability

Not Applicable

Code availability

The full code is available at https://gitlab.ilabt.imec.be/mwbaert/maxcausalent-icrl.

Authors' contributions

All authors contributed to the study conception and design. Material preparation, data collection, coding and analysis were performed by Mattijs Baert. The first draft of the manuscript was written by Mattijs Baert and all authors edited and commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

[1] Christian, B.: The Alignment Problem: Machine Learning and Human Values. WW Norton & Company, UK (2020)

[2] Balakrishnan, A., Bouneffouf, D., Mattei, N., Rossi, F.: Incorporating behavioral constraints in online ai systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 3–11 (2019)

[3] Russell, S.: Human Compatible: Artificial Intelligence and the Problem of Control. Penguin, UK (2019)

[4] Van Moffaert, K., Nowé, A.: Multi-objective reinforcement learning using sets of pareto dominating policies. The Journal of Machine Learning Research **15**(1), 3483–3512 (2014)

[5] Mania, H., Guy, A., Recht, B.: Simple random search provides a competitive approach to reinforcement learning. arXiv preprint arXiv:1803.07055 (2018)

[6] Bhatnagar, S., Lakshmanan, K.: An online actor–critic algorithm with function approximation for constrained markov decision processes. Journal of Optimization Theory and Applications **153**(3), 688–708 (2012)

[7] Tessler, C., Mankowitz, D.J., Mannor, S.: Reward constrained policy optimization. arXiv preprint arXiv:1805.11074 (2018)

[8] Ho, J., Ermon, S.: Generative adversarial imitation learning. Advances in neural information processing systems **29** (2016)

[9] Finn, C., Levine, S., Abbeel, P.: Guided cost learning: Deep inverse optimal control via policy optimization. In: International Conference on Machine Learning, pp. 49–58 (2016). PMLR

[10] Fu, J., Luo, K., Levine, S.: Learning robust rewards with adversarial inverse reinforcement learning. arXiv preprint arXiv:1710.11248 (2017)

[11] Scobee, D.R., Sastry, S.S.: Maximum likelihood constraint inference for inverse reinforcement learning. arXiv preprint arXiv:1909.05477 (2019)

[12] Stocking, K.C., McPherson, D.L., Matthew, R.P., Tomlin, C.J.: Discretizing Dynamics for Maximum Likelihood Constraint Inference (2021). https://arxiv.org/abs/2109.04874

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

[14] Baert, M., Leroux, S., Simoens, P.: Inverse reinforcement learning through logic constraint inference. Machine Learning, 1–26 (2023)

[15] Papadimitriou, D., Anwar, U., Brown, D.S.: Bayesian inverse constrained reinforcement learning. In: Workshop on Safe and Robust Control of Uncertain Systems (NeurIPS) (2021)

[16] Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K., *et al.*: Maximum entropy inverse reinforcement learning. In: Aaai, vol. 8, pp. 1433–1438 (2008). Chicago, IL, USA

[17] Glazier, A., Loreggia, A., Mattei, N., Rahgooy, T., Rossi, F., Venable, B.: Learning Behavioral Soft Constraints from Demonstrations (2022). https://arxiv.org/abs/2202.10407

[18] Malik, S., Anwar, U., Aghasi, A., Ahmed, A.: Inverse constrained reinforcement learning. In: International Conference on Machine Learning, pp. 7390–7399 (2021). PMLR

[19] Liu, G., Luo, Y., Gaurav, A., Rezaee, K., Poupart, P.: Benchmarking constraint inference in inverse reinforcement learning. arXiv preprint arXiv:2206.09670 (2022)

[20] Ziebart, B.D., Bagnell, J.A., Dey, A.K.: Modeling interaction via the principle of maximum causal entropy. In: ICML (2010)

[21] Qiao, G., Liu, G., Poupart, P., *et al.*: Multi-modal inverse constrained reinforcement learning from a mixture of demonstrations. In: Thirty-seventh Conference on Neural Information Processing Systems (2023)

[22] Kim, K., Swamy, G., Liu, Z., Zhao, D., Choudhury, S., Wu, Z.S.: Learning shared safety constraints from multi-task demonstrations (2023)

[23] Baert, M., Leroux, S., Simoens, P.: Learning logic constraints from demonstration. In: NeSy2023: 17th International Workshop on Neural-Symbolic Learning and Reasoning, pp. 78–84 (2023)

[24] Lindner, D., Chen, X., Tschiatschek, S., Hofmann, K., Krause, A.: Learning safety constraints from demonstrations with unknown rewards. arXiv preprint arXiv:2305.16147 (2023)

[25] Gaurav, A., Rezaee, K., Liu, G., Poupart, P.: Learning soft constraints from constrained expert demonstrations. arXiv preprint arXiv:2206.01311 (2022)

[26] Bain, M., Sammut, C.: A framework for behavioural cloning. In: Machine Intelligence 15, pp. 103–129 (1995)

[27] Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 627–635 (2011). JMLR Workshop and Conference Proceedings

[28] Chu, W., Ghahramani, Z.: Preference learning with gaussian processes. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 137–144 (2005)

[29] Lee, K., Smith, L., Dragan, A., Abbeel, P.: B-pref: Benchmarking preference-based reinforcement learning. arXiv preprint arXiv:2111.03026 (2021)

[30] Chou, G., Berenson, D., Ozay, N.: Learning constraints from demonstrations. In: Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13, pp. 228–245 (2020). Springer

[31] Chou, G., Ozay, N., Berenson, D.: Learning parametric constraints in high dimensions from demonstrations. In: Conference on Robot Learning, pp. 1211–1230 (2020). PMLR

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

[33] Chou, G., Berenson, D., Ozay, N.: Uncertainty-aware constraint learning for adaptive safe motion planning from demonstrations. In: Conference on Robot Learning, pp. 1612–1639 (2021). PMLR

[34] Chou, G., Wang, H., Berenson, D.: Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations. IEEE Robotics and Automation Letters **7**(2), 3827–3834 (2022)

[35] Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-first International Conference on Machine Learning, p. 1 (2004)

[36] Altman, E.: Constrained Markov Decision Processes: Stochastic Modeling. Routledge, ??? (1999)

[37] Wulfmeier, M., Ondruska, P., Posner, I.: Maximum entropy deep inverse reinforcement learning. arXiv preprint arXiv:1507.04888 (2015)

[38] Gleave, A., Toyer, S.: A primer on maximum causal entropy inverse reinforcement learning. arXiv preprint arXiv:2203.11409 (2022)

[39] Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge university press, ??? (2004)

[40] Borkar, V.S.: Stochastic Approximation: a Dynamical Systems Viewpoint vol. 48. Springer, ??? (2009)

[41] Haarnoja, T., Tang, H., Abbeel, P., Levine, S.: Reinforcement learning with deep energy-based policies. In: International Conference on Machine Learning, pp. 1352–1361 (2017). PMLR

[42] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3), 229–256 (1992)

[43] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[44] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)

[45] Krajewski, R., Bock, J., Kloeker, L., 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). https://doi.org/10.1109/ITSC.2018.8569552

[46] Wang, X., Krasowski, H., Althoff, M.: Commonroad-rl: A configurable reinforcement learning environment for motion planning of autonomous vehicles. In: IEEE International Conference on Intelligent Transportation Systems (ITSC) (2021). https://doi.org/10.1109/ITSC48978.2021.9564898

[47] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning, pp. 1861–1870 (2018). PMLR

[48] Kostrikov, I., Nachum, O., Tompson, J.: Imitation learning via off-policy distribution matching. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, ??? (2020). https://openreview.net/forum?id=Hyg-JC4FDr

[49] Garg, D., Chakraborty, S., Cundy, C., Song, J., Ermon, S.: Iq-learn: Inverse soft-q learning for imitation. Advances in Neural Information Processing Systems **34**, 4028–4039 (2021)

[50] Chan, A.J., Schaar, M.: Scalable Bayesian inverse reinforcement learning. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=4qR3coiNaIv

[51] Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438 (2015)

[52] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (2018)

[53] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)