Learning Behavioral Soft Constraints from Demonstrations

Arie Glazier
Tulane University
adglazier@gmail.com

Andrea Loreggia
EUI
andrea.loreggia@gmail.com

Nicholas Mattei
Tulane University
nsmattei@tulane.edu

Taher Rahgooy
University of West Florida
trahgooy@students.uwf.edu

Francesca Rossi
IBM Research
Francesca.Rossi2@ibm.com

Brent Venable
IHMC
bvenable@ihmc.org

Abstract

Many real-life scenarios require humans to make difficult trade-offs: do we always follow all the traffic rules or do we violate the speed limit in an emergency? These scenarios force us to evaluate the trade-off between collective norms and our own personal objectives. To create effective AI-human teams, we must equip AI agents with a model of how humans make trade-offs in complex, constrained environments. These agents will be able to mirror human behavior or to draw human attention to situations where decision making could be improved. To this end, we propose a novel inverse reinforcement learning (IRL) method for learning implicit hard and soft constraints from demonstrations, enabling agents to quickly adapt to new settings. In addition, learning soft constraints over states, actions, and state features allows agents to transfer this knowledge to new domains that share similar aspects.

1 Introduction

Implicit and explicit constraints are present in many decision making scenarios, and their presences forces us to make difficult decisions: do we always satisfy all constraints, or do we violate some of them in exceptional circumstances? Many techniques can be used to combine constraints and goals so that an autonomous agent rationally minimizes constraint violations while achieving the given goal [11]. However, it is well known that humans are not rational. When we need to make a decision in a constrained environment, we often reason by employing heuristics and approximations which are subject to bias and noise [5, 6]. This means that optimal techniques may not be suitable if the aim is to design autonomous artificial agents that act like humans, or decision support systems that simulate human behavior to anticipate it and possibly alert humans by making them aware of their reasoning and inference deficiencies.

Moreover, these constraints are often not explicitly given, but need to be inferred from observations of how other agents act in the constrained world. Learning constraints from demonstrations is an important topic in the domains of inverse reinforcement learning [1, 16], which is used to implement AI safety goals including value alignment [4, 8, 15] and to circumvent reward hacking [3, 12]. Recent work has focused on building ethically bounded agents [4] that comply with ethical or moral theories of action [13, 18]. Following the work of Scobee and Sastry [16], we propose an architecture that, given access to a model of the environment and to demonstrations of constrained behavior, is able to learn constraints associated with states, actions, or state features. Our method, MESC-IRL, performs comparably with the state of the art and is more general, as it can handle both hard and soft constraints

Workshop on Safe and Robust Control of Uncertain Systems at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Online.

in both deterministic and non-deterministic environments. It is also decomposable into features of the environment, supporting the transfer of learned constraints between environments.

Contributions. We propose and evaluate a novel method, MESC-IRL, that is able to learn both hard and soft constraints in both deterministic and non-deterministic MDPs from a set of demonstrations. This method strictly generalizes existing methods in the literature and achieves state of the art performance. Our method is also decomposable into features of the environment, which supports transferring learned constraints between environments.

2 Constrained MDPs and Inverse Reinforcement Learning

We begin this section by providing the preliminary notions on the context of our work, that is, constrained Markov Decision Processes and Reinforcement Learning [17]. We then review fundamental concepts and methods on Inverse Reinforcement Learning [1, 10] and background on Constrained Markov Decision Processes [2] including related work on learning constraints [9, 16, 20], which we will leverage to develop our novel method for learning soft constraints [14] from demonstrations [7].

3 Markov Decision Processes and Reinforcement Learning

A finite-horizon Markov Decision Process (MDP) $\mathcal{M}$ is a model for sequential decision making over a number of time steps $t \in T$ defined by a tuple $(\mathcal{S}, \mathcal{A}, P, D_0, \phi, \gamma, R)$ [17]. $\mathcal{S}$ is a finite set of discrete states; $\{\mathcal{A}_s\} \subseteq \mathcal{A}$ is a set of actions available at state s; $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a model of the environment given as transition probabilities where $P(s_{t+1}|s_t, a_t)$ is the probability of transitioning to state s_{t+1} from state s_t after taking action $a_t \in \{\mathcal{A}_{s_t}\}$ at time t. $D_0 : \mathcal{S} \to [0, 1]$ is a distribution over start states; $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^k$ is a mapping from the transitions to a k-dimensional space of features; $\gamma \in [0, 1)$ is a discount factor; and $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a scalar reward received by the agent for being in one state and transitioning to another state at time t, written as $R(s_t, a_t, s_{t+1})$.

An agent acts within the environment defined by the MDP, generating a sequence of actions called a *trajectory* of length t. Let $\tau = ((s_1, a_1, s_2), ..., (s_{t-1}, a_{t-1}, s_t)) \in (\mathcal{S} \times \mathcal{A} \times \mathcal{S})^t$. We evaluate the quality of a particular trajectory in terms of the amount of reward accrued over the trajectory, subject to discounting. Formally, $R(\tau) = \sum_{i=1}^{t} \gamma^i R(s_i, a_i, s_{i+1})$. A policy, $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ is a map of probability distribution to actions for every state such that $\pi(s, a)$ is the probability of taking action a in state s. We can also write the probability of a trajectory τ under a policy as $\pi(\tau)$. The feature vector associated with trajectory τ is defined as the summation over all transition feature vectors in τ, $\phi(\tau) = \sum_{(s_t, a_t, s_{t+1}) \in \tau} \phi(s_t, a_t, s_{t+1})$

The goal within an MDP is to find a policy π^* that maximizes the expected reward, $J(\pi) = \mathbb{E}_{\tau \sim \pi}[R(\tau)]$ [9]. In the MDP literature, classical tabular methods are used to find π^* including value iteration (VI). Such method finds an optimal policy by estimating the expected reward for taking an action a in a given state s, i.e., the Q-value of pair (s, a), written $q(s, a)$. [17].

3.1 Constrained MDPs and Inverse Reinforcement Learning

We are interested in learning constraints from a set of demonstrations $\mathcal{D}$. Our goal is to create agents that are able to be trained to follow constraints that are not explicitly prohibited in the MDP, but should be avoided [13]. [16] discusses the importance of such constraints: an MDP $\mathcal{M}$ may encode everything necessary about driving a car, e.g. the dynamics of steering and movements, but often one wants to add additional general constraints such as *avoid obstacles on the way to the goal*. These constraints are often non-Markovian and engineering a reward function that encodes these constraints may be a difficulty or impossible task [19].

One approach for learning constraints from demonstrations is to use techniques from inverse reinforcement learning (IRL): given a set of demonstrated trajectories $\mathcal{D}$ of an agent in an environment $\mathcal{M}$ with an unknown reward function $\mathcal{M} \setminus R$, IRL provides a set of techniques for learning a reward function $\hat{R}$ that explains the agent's demonstrated behavior [1, 10]. However, this technique has many drawbacks: often there are many reward functions that lead to the same behavior [16], the reward functions may not be interpretable [19], and there may be issues such as reward hacking –

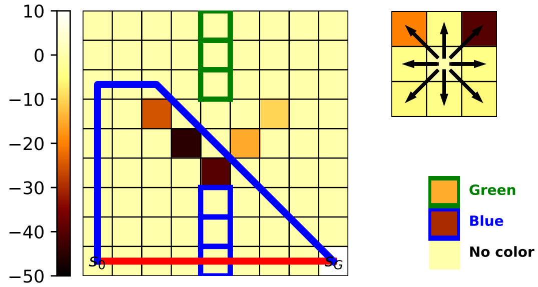

Figure 1: Example grid with constraints of varying costs over actions, state occupancy, and state features. Note $\pi^*_{\mathcal{N}}$ for $\mathcal{M}^{\mathcal{N}}$ is the red trajectory, obtained by an agent that does not know the constraints while $\pi^*_{\mathcal{C}^*}$ for $\mathcal{M}^{\mathcal{C}^*}$ is the blue trajectory.

wherein the agent learns to behave in ways that create reward but are not intended by the designer – an important topic in the field of AI safety [3, 12] and value alignment [13, 15].

We follow the framework of Altman [2] and Malik et al. [9] and define a Constrained MDP $\mathcal{M}^{\mathcal{C}}$ which is a nominal MDP $\mathcal{M}^{\mathcal{N}}$ with an additional cost function $\mathcal{C} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and a budget $\alpha \geq 0$. We can then define the cost of a trajectory to be $c(\tau) = \sum_{i=1}^{t} c(s_i, a_i, s_{i+1})$. Setting $\alpha = 0$ is enforcing *hard constraints*, i.e., we must never trigger constrained transitions. In this work, unlike the work of both Scobee and Sastry [16] and Malik et al. [9], we are interested in learning *soft constraints* [14]. Under a soft constraints paradigm, each constraint comes with a real-valued penalty/cost and the goal is to minimize the sum of penalties incurred by the agent.

Following Scobee and Sastry [16], the task of constraint inference in IRL is defined as follows. Given a nominal MDP $\mathcal{M}^{\mathcal{N}}$ and a set of demonstrations $\mathcal{D}$ in ground-truth constrained world $\mathcal{M}^{\mathcal{C}^*}$, we wish to find the most likely set of constraints $\mathcal{C}$ that could modify $\mathcal{M}^{\mathcal{N}}$ to explain the demonstrations. We are concerned with three types of constraints:

Action Constraints: We may not want an agent to ever perform some (set of) action a_i;

Occupancy Constraints: We may not want an agent to occupy a (set of) states s_i;

Feature Constraints: Given a feature mapping of transitions ϕ, we may not want an agent to perform an (set of) action in presence of specific state features.

Without loss of generality, we add the state and actions to the features. Hence, action and occupancy become specific cases of feature constraints. Note that the set of constraints is defined as a cost function over the set of transitions $\mathcal{C} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. In Scobee and Sastry [16], this definition is limited to a set of state-actions $\mathcal{S} \times \mathcal{A}$ as they are assuming a deterministic setting and hence are able to define $\mathcal{M}^{\mathcal{C}}$ by substituting $\mathcal{A} = \{A_s\}$ with $\mathcal{A}^{\mathcal{C}} = \{A_s^{\mathcal{C}}\}$ in $\mathcal{M}^{\mathcal{N}}$. Finally, both Scobee and Sastry [16] and Malik et al. [9] propose a greedy approach to infer a set of constraint $\mathcal{C}$ that explains the demonstrations D on $\mathcal{M}^{\mathcal{C}^*}$. In both Scobee and Sastry [16] and Malik et al. [9] the domain is restricted to deterministic MDPs, which we strictly generalize in this work; additionally Scobee and Sastry [16], like our model, only works with discrete actions, while Malik et al. [9] works for both discrete and continuous action sets. We also generalize to the non-deterministic setting; we additionally generalize to the setting of soft constraints, hence our task is to learn the cost function $\mathcal{C}$.

To test our methods, we use the same grid world setup as Scobee and Sastry [16]. Within our grid world example, shown in Figure 1 (Left), we have an action penalty of -4 for the cardinal directions, $-4 \times \sqrt{2}$ for taking the diagonal actions, and reaching the goal state has a reward of 10. In Figure 1 (Left) we set the constraint costs to various values but in all our experiments we fix the constraint costs on the generated grids for states, actions, and features to be -50. The feature vector ϕ is a one-hot vector representing the color associated with each state: blue, green, and no color. Throughout we assume a non-deterministic world with a 10% chance of action failure, resulting in a random action.

4 MESC-IRL: Max Entropy Inverse Soft-Constraint RL

We now describe our method for learning a set soft constraints from a set of demonstrations $\mathcal{D}$ and a nominal MDP $\mathcal{M}^{\mathcal{N}}$. Our method described here generalizes the work of both Scobee and Sastry [16] and Malik et al. [9] to the setting of non-deterministic MDPs and soft constraints. Following Ziebart et al. [20], our goal is to optimize a function that linearly maps the features of each transition to the reward associated with that transition, $R(s_t, a_t, s_{t+1}) = \omega\phi(s_t, a_t, s_{t+1})$, where ω is the reward weight vector. Ziebart et al. [20] propose a maximum entropy model for finding a unique solution (ω) for this problem. Based on this model, the probability of finite-length trajectory τ being executed by an agent traversing an MDP $\mathcal{M}$ is exponentially proportional to the reward earned by that trajectory and can be approximated by $P(\tau|\omega) \approx \frac{e^{\omega^T\phi(\tau)}}{Z(\omega)} \prod_{(s_t,a_t,s_{t+1})\in\tau} P(s_{t+1}|s_t, a_t)$. The optimal solution is obtained by finding the maximum likelihood of the demonstrations $\mathcal{D}$ using this probability distribution: $\omega^* = \underset{\omega}{\operatorname{argmax}} \sum_{\tau\in\mathcal{D}} \log P(\tau|\omega)$.

We extend the setting of Scobee and Sastry [16] to learning a set of *soft* constraints which best explain $\mathcal{D}$. Allowing us to move from the notion of a constraint forbidding an action or a state to a soft constraint imposing a penalty proportional to the gravity of its violation. Given access to $\mathcal{M}^{\mathcal{N}}$ and a set of demonstrations $\mathcal{D}$ in ground-truth constrained MDP $\mathcal{M}^{\mathcal{C}^*}$ we want to find the costs $\mathcal{C}$. Formally, we define the residual reward function $R^{\mathcal{R}} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ as a mapping from the transitions to the penalties. We can now formally define our soft-constrained MDP $\mathcal{M}^{\mathcal{C}}$ as follows: Given $\mathcal{M}^{\mathcal{N}} = \langle \mathcal{S}, \mathcal{A}, P, \mu, \phi, R^{\mathcal{N}} \rangle$ we define *soft-constrained MDP* $\mathcal{M}^{\mathcal{C}} = \langle \mathcal{S}, \mathcal{A}, P, \mu, \phi, R^{\mathcal{C}} \rangle$ where $R^{\mathcal{C}} = R^{\mathcal{N}} - R^{\mathcal{R}}$. Thus, the goal of our task is to find a residual reward function $R^{\mathcal{R}}$ that maximizes the likelihood of the demonstrations $\mathcal{D}$ given the nominal MDP $\mathcal{M}^{\mathcal{N}}$.

Our solution is based on adapting Maximum Causal Entropy Inverse Reinforcement learning [20, 21] to soft-constrained MDPs. Following the setting of Ziebart et al. [20] we can write the reward function $R^{\mathcal{N}}$ (resp. $R^{\mathcal{C}}$) of $\mathcal{M}^{\mathcal{N}}$ (resp. $\mathcal{M}^{\mathcal{C}}$) as a linear combination of the transitions: $R^{\mathcal{N}}(s_t, a_t, s_{t+1}) = \omega^{\mathcal{N}}\phi(s_t, a_t, s_{t+1})$ and $R^{\mathcal{C}}(s_t, a_t, s_{t+1}) = \omega^{\mathcal{C}}\phi(s_t, a_t, s_{t+1})$. As, both reward functions $R^{\mathcal{N}}$ and $R^{\mathcal{C}}$ are linear, $R^{\mathcal{R}}$ should be linear as well: $R^{\mathcal{R}} = \omega^{\mathcal{R}}\phi(s_t, a_t, s_{t+1})$. From this formulation of $R^{\mathcal{R}}$ we can infer that the reward vectors follow $\omega^{\mathcal{C}} = \omega^{\mathcal{N}} - \omega^{\mathcal{R}}$.

We can use Max Entropy IRL for learning a reward function compatible with the set $\mathcal{D}$. The gradient for maximizing the likelihood in this setting is defined as in Ziebart et al. [20]: $\nabla_{\omega^{\mathcal{C}}}\mathcal{L}(\mathcal{D}) = \mathbb{E}_{\mathcal{D}}[\phi(\tau)] - \sum_{(s_t,a_t,s_{t+1})} D_{s_t,a_t,s_{t+1}}\phi(s_t, a_t, s_{t+1})$. Where $D_{s_t,a_t,s_{t+1}}$ is the expected feature frequencies for transition (s_t, a_t, s_{t+1}) using the current $\omega^{\mathcal{C}}$ weights. As the reward vectors follow $\omega^{\mathcal{C}} = \omega^{\mathcal{N}} - \omega^{\mathcal{R}}$, we have $\nabla_{\omega^{\mathcal{C}}} = -\nabla_{\omega^{\mathcal{R}}}$. Finally, by substituting this in the above, we obtain the gradient of likelihood of the constrained trajectories w.r.t. $\omega^{\mathcal{R}}$: $\nabla_{\omega^{\mathcal{R}}}\mathcal{L}(\mathcal{D}) = \sum_{(s_t,a_t,s_{t+1})} D_{s_t,a_t,s_{t+1}}\phi(s_t, a_t, s_{t+1}) - \mathbb{E}_{\mathcal{D}}[\phi(\tau)]$. As we estimate the residual rewards w.r.t. the nominal rewards, these rewards are automatically scaled to be compatible with the nominal rewards.

5 Generalizing From Penalties to Probabilities

The estimated penalties from the previous section can effectively guide an agent to navigate the environment optimally as well as provide estimates of the cost of the constraints scaled to the value of the original reward signal. However, there may be instances, such as when comparing with hard constraints, where we desire *probabilities* that a particular action is constrained. Having probabilities allows us to compare constraints across environments with possibly different scales, allows us to use this information to guide our policies, and allows us to evaluate the confidence we have in a particular constraint. In this section we describe a method to transition from penalties to probabilities, as well as a generalized method to extract these probabilities based on a subset of the features of the environment, which can facilitate transfer learning between domains.

A transition where the residual reward, i.e., the penalty, is significantly larger than zero is more likely to be a constraint. We estimate the significance of a penalty by scaling it to the standard deviation of the mean learned reward. Therefore, we assume that a transition penalty is a random variable, denoted by $\mathbb{C} \sim logistic(\sigma_{pooled}, \sigma_{pooled})$, following a logistic distribution with standard deviation σ_{pooled}, where $\sigma_{pooled} = \sqrt{(\sigma_{\mathcal{N}}^2+\sigma_{\mathcal{C}}^2)/2}$ and $\sigma_{\mathcal{N}}$ and $\sigma_{\mathcal{C}}$ are the standard deviations of the rewards in

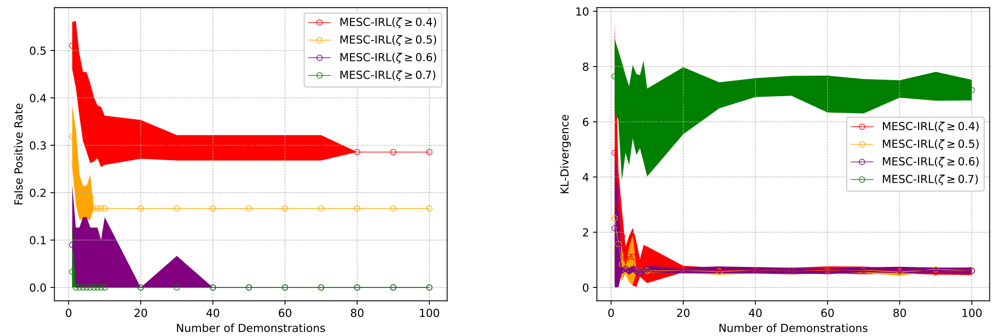

Figure 2: Performance of MESC-IRL for various settings of ζ at recovering hard constraints in a deterministic setting according to false positive rate (left) and KL-Divergence from the demonstrations $\mathcal{D}$ (right) as we vary the number of demonstrations. Each point is the mean of 10 independent draws.

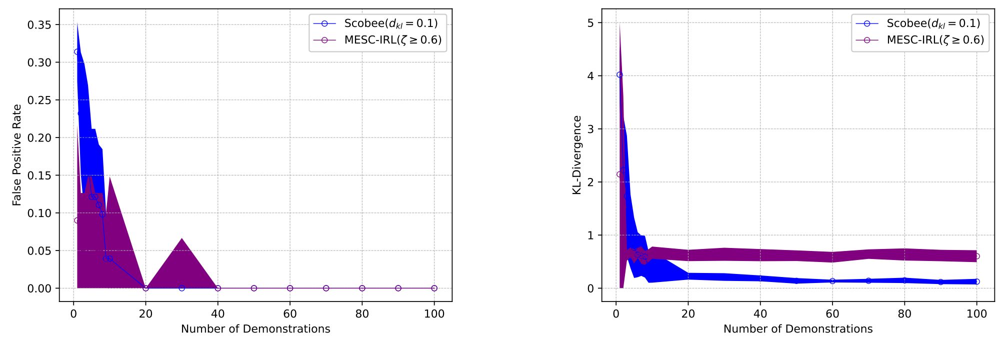

Figure 3: Comparision of the best performing version of MESC-IRL to the best performing method of Scobee and Sastry [16] at recovering hard constraints in a deterministic setting according to false positive rate (left) and KL-Divergence from the demonstrations $\mathcal{D}$ (right) as we vary the number of demonstrations. Each point is the mean of 10 independent draws.

the nominal and learned constrained worlds, respectively. When penalties are close to zero, we want their probabilities to be small. To do this we set the mean of the distribution to be $\mu = \sigma_{pooled}$.

We now want to reason about a random variable ζ that indicates our belief that the transition (s_t, a_t, s_{t+1}) is forbidden. Hence using the above probability distribution we can define the probability of constraint given a transition as:

$$\zeta \equiv P\left(\mathbb{C} \leq R^R(s_t, a_t, s_{t+1})\right) = sigmoid\left({R^R(s_t,a_t,s_{t+1}) - \sigma_{pooled}}/{\sigma_{pooled}} \right).$$

In our formulation, the residual rewards only depend on the features associated with them. Hence, we can use this fact to reason about constraints over only a subset of features $\mathbf{f}$, e.g., only color or state position. Let $\phi_f \subseteq \phi$ be the subset of features we are concerned with. In our grids we represent ϕ with a vector of length 92. The first 81 elements represent the states, the next 8 represent the actions, and the last 3 represent the colors. So if we are interested in only learning about constraints over the colors, ϕ_f will be a vector equal to the last three elements of ϕ that is $\phi_{color} \equiv \phi_{90,91,92}$.

Let $\phi_{\mathbf{f}}$ and $\omega^R_{\mathbf{f}}$ be the feature function and residual feature weight vector for $\mathbf{f}$. We can now define the probability of a feature value to be constrained as:

$$\zeta_{\mathbf{f}} \equiv P(\mathbb{C} \leq \omega^R_{\mathbf{f}} \phi_{\mathbf{f}}) = sigmoid\left({\omega_{\mathbf{f}} \phi_{\mathbf{f}} - std_{pooled}}/{std_{pooled}} \right).$$

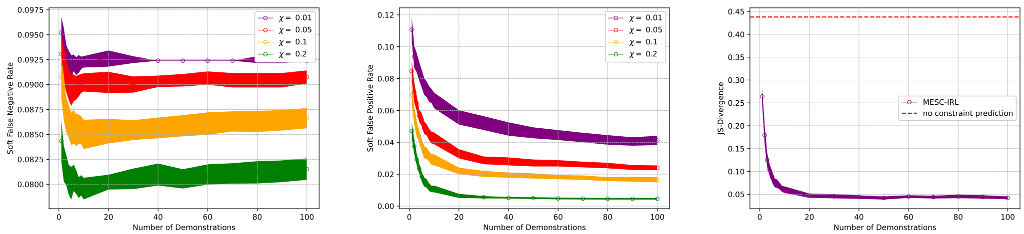

Figure 4: Performance of MESC-IRL on recovering soft constraints in deterministic settings according to false negatives (left), false positives (center), and JS-Divergence to $\mathcal{D}$ (right). We see that across all these settings we are able to accurately recover constraints and generate behavior similar to the $\mathcal{D}$ even with few demonstrations.

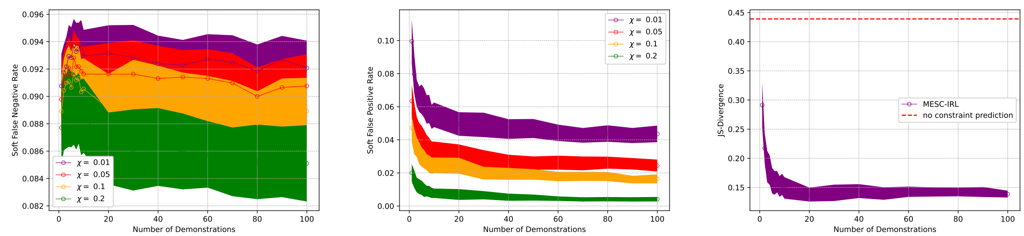

Figure 5: Performance of MESC-IRL on recovering soft constraints in deterministic settings according to false negatives (left), false positives (center), and JS-Divergence to $\mathcal{D}$ (right). We see that across all these settings we are able to accurately recover constraints and generate behavior similar to the $\mathcal{D}$ even with few demonstrations.

6 Experimental Evaluation of MESC-IRL

In this section we empirically validate our method for soft constraint learning against both the method of Scobee and Sastry [16] for learning hard constraints in deterministic settings as well as on learning soft constraints in non-deterministic settings. Figures 2 and 3 shows the performance of MESC-IRL compared to the method proposed by Scobee and Sastry [16] on the same metrics from their paper: false positives, i.e., predicting a constraint when one does not exist, and KL-Divergence from the demonstrations set $\mathcal{D}$. For this test we use the same single grid, hard constraints, and a deterministic setting to allow for a direct comparison. We generate 10 independent sets of 100 demonstrations and report the mean. In order to decide if the values returned by MESC-IRL represent a hard constraint, we threshold the value of ζ at various levels and plot the comparison to the best result from Scobee and Sastry [16]. MESC-IRL with $\zeta \geq 0.6$ performs better than existing methods when the number of demonstrations is low, about the same when there are more demonstrations, and is able to also work for soft constraints and non-deterministic settings.

To evaluate MESC-IRL on soft constraints we need to adapt the notion of false positives and false negatives. Let a false positive fp be:

$$fp = \frac{\left|\left\{x \in \mathcal{C} \mid c(x) = 0 \land (\zeta_{\mathcal{C}}(x) - \zeta_{\mathcal{C}^*}(x) > \chi)\right\}\right|}{\text{Num. Constraints}},$$

where $\zeta_{\mathcal{C}}(x)$ and $\zeta_{\mathcal{C}^*}(x)$ are the predicted and true probability of transition x being constrained as described in Section 5, and χ is a value in [0,1]. Intuitively, we count a constraint as a false positive whenever there is no constraint in $\mathcal{M}^{\mathcal{C}^*}$ and the predicted probability exceeds the true probability by more than the threshold χ. We can adapt the notion of false negatives, fn in the same way by taking $c(x) \neq 0$.

Figures 4 and 5 shows the results of our tests on recovering soft constraints in both deterministic and non-deterministic settings with random grids. For these tests we choose a start and a goal state randomly at least 8 moves apart, set 6 states for blue, 6 for green randomly, and select 6 randomly constrained states; all penalties are set to -50. Again we take 10 sets of 100 demonstrations. We see a strong decrease in both false positives and false negatives as the number of demonstrations grows. We see that in general, and even more so when the optimal threshold $\chi = 0.2$ is selected, our method almost never adds constraints that are not present in the ground truth and rarely underestimates the probability of existing ones, even for small demonstration sets. Likewise our method is able to generate trajectories very close to $\mathcal{D}$, showing that we are able to recover both constraints even with soft constraints in non-deterministic setting. Hence MESC-IRL is able to work across a variety of settings and accurately capture demonstrated constraints.

Acknowledgements

Nicholas Mattei was supported by NSF Award IIS-2007955 and an IBM Faculty Research Award. K. Brent Venable are supported by NSF Award IIS-2008011.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

[2] Eitan Altman. *Constrained Markov Decision Processes*, volume 7. CRC Press, 1999.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[4] Avinash Balakrishnan, Djallel Bouneffouf, Nicholas Mattei, and Francesca Rossi. Incorporating behavioral constraints in online ai systems. In *Proc. of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[5] Grady Booch, Francesco Fabiano, Lior Horesh, Kiran Kate, Jonathan Lenchner, Nick Linck, Andrea Loreggia, Keerthiram Murugesan, Nicholas Mattei, Francesca Rossi, and Biplav Srivastava. Thinking fast and slow in AI. In *Proc. of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pages 15042–15046, 2021.

[6] Jerome R Busemeyer and Adele Diederich. Survey of decision field theory. *Mathematical Social Sciences*, 43(3):345–370, 2002.

[7] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. *arXiv preprint arXiv:1812.07084*, 2018.

[8] Andrea Loreggia, Nicholas Mattei, Francesca Rossi, and K. Brent Venable. Preferences and ethical principles in decision making. In *Proc. 1st ACM/AAAI Conference on AI, Ethics & Society (AIES)*, 2018.

[9] Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pages 7390–7399. PMLR, 2021.

[10] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 663–670, 2000.

[11] Ritesh Noothigattu, Djallel Bouneffouf, Nicholas Mattei, Rachita Chandra, Piyush Madan, Kush R. Varshney, Murray Campbell, Moninder Singh, and Francesca Rossi. Teaching AI agents ethical values using reinforcement learning and policy orchestration. *IBM J. Res. Dev.*, 63(4/5):2:1–2:9, 2019.

[12] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7, 2019.

[13] Francesca Rossi and Nicholas Mattei. Building ethically bounded AI. In *Proc. of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[14] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of Constraint Programming*. Elsevier, 2006.

[15] S. Russell, D. Dewey, and M. Tegmark. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4):105–114, 2015.

[16] Dexter R. R. Scobee and S. Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. In *8th International Conference on Learning Representations ICLR*. OpenReview.net, 2020.

[17] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, 2nd Edition*. A Bradford Book, Cambridge, MA, USA, 2018.

[18] Justin Svegliato, Samer B Nashed, and Shlomo Zilberstein. Ethically compliant sequential decision making. In *Proceedings of the 35th AAAI International Conference on Artificial Intelligence (AAAI)*, 2021.

[19] Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K. Ho, and Sanjit A. Seshia. Learning task specifications from demonstrations. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Annual Conference on Neural Information Processing Systems 2018 (NeurIPS 2018)*, pages 5372–5382, 2018.

[20] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pages 1433–1438. AAAI Press, 2008.

[21] Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. Modeling interaction via the principle of maximum causal entropy. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1255–1262. Omnipress, 2010.