# Maximum Likelihood Constraint Inference on Continuous State Spaces

**Kaylene C. Stocking**[1], **D. Livingston McPherson**[1], **Robert P. Matthew**[2], **Claire J. Tomlin**[1]
1 University of California, Berkeley: `{kaylene, david.mcpherson, tomlin}@berkeley.edu`
2 University of California, San Francisco: `robert.matthew@ucsf.edu`

## Abstract

When a robot observes another agent unexpectedly modifying their behavior, inferring the most likely cause is a valuable tool for maintaining safety and reacting appropriately. In this work, we present a novel method for inferring constraints that works on continuous, possibly sub-optimal demonstrations. We first learn a representation of the continuous-state maximum entropy trajectory distribution using deep reinforcement learning. We then use Monte Carlo sampling from this distribution to generate expected constraint violation probabilities and perform constraint inference. When the agent's dynamics and objective function are known in advance, this process can be performed offline, allowing for real-time constraint inference at the moment demonstrations are observed. We demonstrate our approach on two continuous systems, including a human driving a model car.

## 1 Introduction

The behavior of an agent is an important source of information about their goals and surroundings. For example, inverse reinforcement learning estimates the reward function an agent appears to be optimizing [12]. In many practical situations, however, we already have a good idea of the reward function. In cases like these, an agent's actions can still provide useful information. Consider a driver who temporarily swerves out of their lane to avoid an obstacle. Even if we can't directly observe the obstacle, a reasonable inference is that something is preventing the driver from taking the expected path. The driver's swerving behavior is surprising if no obstacle is present, but becomes much more likely when we incorporate a possible obstacle into our model. This intuition can be formalized as maximum likelihood constraint inference (MLCI), first developed in [10]. MLCI uses the maximum entropy framework [12] to identify which constraint in a constraint hypothesis set provides the best explanation for unexpected demonstrator behavior. Constraint inference may be an especially useful form of unsupervised learning for robots, which can use identified constraints to maintain their own safety even when performing a different task than the demonstrator.

Unfortunately, the MLCI algorithm presented in [10] only works for system models with discrete, finite state-action spaces. In this work, we extend MLCI to continuous state spaces. Our method retains several key advantages from [10], including working with sub-optimal demonstrations and allowing for most calculations to be pre-computed before any constrained demonstrations are observed.

## 2 Related Work

Our work is most directly inspired by the maximum likelihood method introduced by [10], which identifies the most likely constraint(s) from a hypothesis set, but only works with discrete state-action spaces. Many other methods work directly with continuous dynamics, but drop the maximum likelihood feature. Chou et al. [2] assume that all possible trajectories that could earn higher reward than the demonstration must be constrained in some way. Other methods use heuristics about the demonstrator's behavior rather than an explicit reward function [8, 4, 6]. Anwar et al. [1] recently presented an approach that works with continuous state-action spaces and retains the maximum likelihood framework by using deep reinforcement learning to identify a constraint function over the state-action space. The method we propose here similarly uses deep RL to handle continuous state spaces, but focuses instead on learning constraints from a pre-specified constraint hypothesis set. One key outcome of this difference in perspective is that our method allows for pre-computation before observing a demonstration, enabling a robot to react to a newly identified constraint in real-time.

## 3 Constraint Inference On Continuous State Spaces

### 3.1 Model Preliminaries

Our approach relies on a system model formulated as a deterministic Markov Decision Process (MDP). The MDP is a tuple of four elements: ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $R$), where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of possible actions to take at each state, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition kernel, and $R : \Xi \to \mathbb{R}$ is the reward function. Each trajectory $\xi$ is a sequence ($s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$) for $t \in [0 : T]$, and the space of all possible trajectories is $\Xi$.

$\mathcal{S}$ and $\mathcal{A}$ can be either discrete or continuous; if both are discrete and finite, we say that the MDP is tabular. The MLCI approach developed by [10] only works for tabular MDPs. The method we propose in section 3.3, however, admits MDPs with discrete or continuous state spaces. Note that the MDP formulation requires discrete time steps; however, continuous dynamics can be approximated well with a sufficiently small discrete time step using standard simulation methods [7].

### 3.2 Maximum Entropy Likelihood on Trajectories

After observing a demonstrator's behavior, we suspect that they may be avoiding a constraint when their trajectory is surprising, in the sense of accumulating significantly less reward than should be possible without a constraint. We can formalize this intuition by defining a distribution of expected demonstrator behavior. Following [12], we adopt the maximum entropy likelihood distribution, under which the probability density of continuous-space trajectories is defined as:

$$\pi(\xi) = \frac{e^{\beta R(\xi)}}{\int_{\xi' \in \Xi} e^{\beta R(\xi')} d\xi'} = \frac{1}{Z} q(\xi) \tag{1}$$

where $\beta$ is a temperature parameter that reflects how noisy the demonstrator is, $q(\xi) = e^{\beta R(\xi)}$ is the unnormalized trajectory probability, and $Z = \int q(\xi) d\xi$ is a normalizing constant. This distribution is defined analogously for discrete state-action spaces, where $Z$ is a sum instead of an integral.

Constraints invalidate any trajectory that would otherwise enter the constrained region of the state-action space, rendering its probability zero. Therefore, when we augment a deterministic MDP with a constraint region $c$, we obtain a new maximum entropy distribution:

$$\pi_c(\xi) = \begin{cases} \frac{1}{Z_c} q(\xi), \ (s_t, a_t) \notin c \ \forall t \in [0 : T] \\ 0, \text{otherwise} \end{cases} \tag{2}$$

Where $Z_c$ is a new, smaller normalizing constant that reflects the trajectory probability mass removed by imposing the constraint. Let $Z_0$ be the normalizing constant for the original, unconstrained MDP. Assuming the demonstrator doesn't violate $c$, their behavior is more likely under the new distribution because $Z_c < Z_0$. In fact, given a uniform belief prior over each constraint in a constraint hypothesis set $\mathcal{C}$, the most likely constraint is the one that yields the smallest $Z_c$. Therefore, to perform MLCI, it is sufficient to calculate the ratio $Z_{c_i}/Z_0$ for each $c_i \in \mathcal{C}$ and identify the smallest value. If subsequent demonstrations violate other $c_i$'s, these are simply removed from the hypothesis set, and the smallest remaining $Z_{c_i}/Z_0$ indicates the likeliest constraint.

**Theorem 1** *Let $\mathbb{1}_{c_i}(\xi)$ be the indicator function that trajectory $\xi$ does not violate constraint $c_i$ at any $t \in [0 : T]$. We have:*

$$\frac{Z_{c_i}}{Z_0} = \mathbb{E}_{\xi \sim \pi_0} [\mathbb{1}_{c_i}(\xi)]$$

(A proof is provided in the supplementary material.) Theorem 1 suggests that a MLCI algorithm requires two components: a representation of the unconstrained maximum entropy distribution $\pi_0(\xi)$, and a way to calculate expected constraint violation.

### 3.3 A Monte Carlo Method for Continuous MLCI

#### 3.3.1 Deep Reinforcement Learning of $\pi_0(\xi)$

As we will see in the following section, a direct representation of the maximum entropy distribution $\pi_0(\xi)$ isn't necessary for MLCI; simply being able to sample trajectories from this distribution is sufficient. This motivates us to approximate $\pi_0(\xi)$ by leveraging deep reinforcement learning (RL). Following [1], consider the KL-divergence between the true maximum entropy distribution $\pi_0(\xi)$ and a learned policy $\pi_\theta(\xi)$ parameterized by $\theta$.

$$\begin{aligned} D_{KL}(\pi_\theta || \pi_0) &= \mathbb{E}_{\xi\sim\pi_\theta}[\log\pi_\theta(\xi) - \log\pi_0(\xi)] \\ &= \mathbb{E}_{\xi\sim\pi_\theta}[\log\pi_\theta(\xi) - \beta R(\xi) + \log Z_0] \end{aligned} \tag{3}$$

Since log $Z_0$ is a constant, we can minimize the divergence by maximizing the following modified expected reward (or objective) function:

$$J(\pi_\theta) = \mathbb{E}_{\xi\sim\pi_\theta}[R(\xi) - \frac{1}{\beta}\log\pi_\theta(\xi)] \tag{4}$$

The parameterization of $\pi_\theta$ is an important limiting factor in how good of an approximation can be achieved. Most RL algorithms working with continuous action spaces learn a Gaussian distribution over actions at each state (e.g. see [3]). However, this is unsuitable for our application because it results in a policy that commits to a single "global" strategy rather than simultaneously exploring multiple strategies with probability determined by their expected reward. Therefore, we instead grid the continuous range of possible action inputs into a few discrete options. A categorical action policy can then be used to learn action distributions with an arbitrary shape at each state. When a short time step $\delta t$ is used, the discretization of the action input only slightly limits the space of policies that can be learned, because switching between discrete values closely approximates any continuous value.

#### 3.3.2 Sampling-Based Approximation of $Z_{c_i}/Z_0$

The learned policy $\pi_\theta$ doesn't directly yield estimates of the trajectory probability distribution $\pi_0(\xi)$. However, for a potential constraint $c_i$ of interest, we can obtain a Monte Carlo estimate of $\mathbb{E}_{\xi\sim\pi_0}[\mathbb{1}_{c_i}(\xi)]$ by simulating policy executions many times and checking whether each sampled trajectory $\xi \sim \pi_\theta$ violates $c_i$. Since the constraint avoidance indicator $\mathbb{1}_{c_i}(\xi)$ for $\xi \sim \pi_\theta$ is a Bernoulli random variable, Hoeffding's inequality tells us that the Monte Carlo estimate will deviate from the true value by more than $\epsilon$ with exponentially decreasing probability $2e^{-2\epsilon^2 n}$ with number of samples $n$. By Theorem 1, calculating the fraction of trajectories that violate each hypothesized constraint is sufficient for determining the most likely constraint.

Neither learning $\pi_\theta$ nor sampling trajectories from this policy relies on observing any demonstrations. Therefore, when the demonstrator's objective function and the constraint hypothesis set are known in advance, both steps can be pre-computed. At inference time, the set of constraints violated by the demonstrator is used to rule these out from the hypothesis set, and the remaining constraint with the highest expected violation (smallest $Z_{c_i}/Z_0$) is the most likely constraint.

## 4 Experiments

### 4.1 Simulated Pendulum System

In the pendulum system, an agent attempts to navigate from a random starting state (angle $\theta$ and angular velocity $\dot{\theta}$) to a random goal state while minimizing control effort and avoiding a ground-truth constraint, either $C_1$ or $C_2$ (see Figure 1a). Both the demonstrator and the inference algorithm use the same simulated dynamics and objective function. To analyze constraint inference performance on this system, we sample demonstrations at random and rank the most likely constraints across 200 trials. We compare the performance of our algorithm to an approximated discrete state space

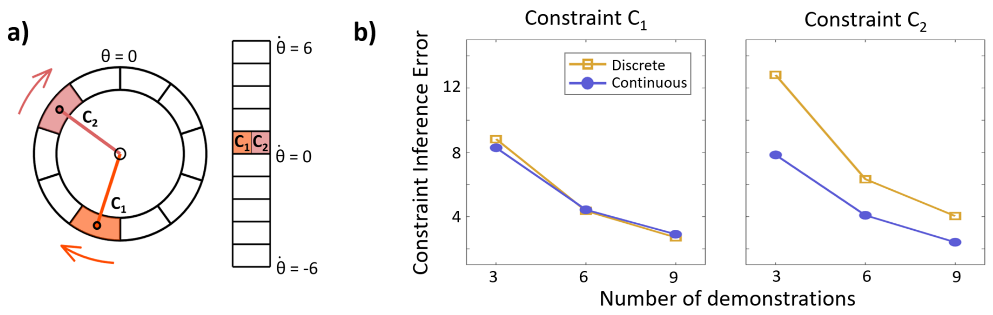

Figure 1: Constraint inference on pendulum dynamics. a) The constraint hypothesis space and ground truth constraints ($C_1$ or $C_2$). The constraint hypothesis space evenly divides the state space into 100 cells. b) Constraint inference performance of our proposed method ("Continuous"), compared to a discretized state space baseline method ("Discrete"). Our method is better at inferring $C_2$, where the dynamics are unstable.

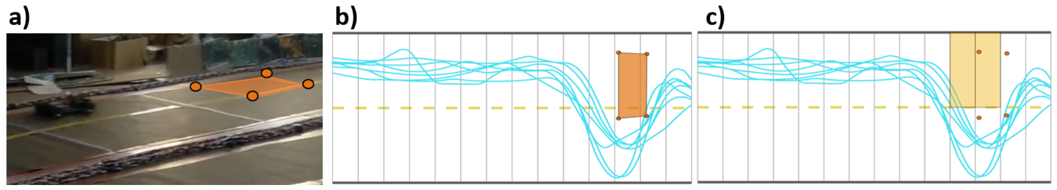

Figure 2: Constraint inference on a 1/10-scale car trajectory along a racetrack. a) The car, piloted by a human, approaches the highlighted obstacle. b) All demonstration trajectories (cyan) and true obstacle region (orange) overlaid on constraint hypothesis set (empty boxes). c) One of two possible constraints is inferred (yellow) after observing a single demonstration.

approach [11] in Figure 1b. Our method identifies the ground truth constraint as one of the most likely after observing only a few demonstrations for both $C_1$ and $C_2$. The inference accuracy of our method is similar to the baseline for $C_1$, but better for $C_2$, where the pendulum dynamics are unstable and the discrete state space approximation suffers accordingly. This difference highlights the advantage of using a continuous state space for performing constraint inference on continuous dynamical systems.

### 4.2 Model Car Remote-Controlled by a Human Driver

To demonstrate the effectiveness of our method in a real-world setting, we performed constraint inference on demonstrations from a human radio-controlling a 1/10 scale model car along a racetrack, attempting to stay in their lane and reach the goal quickly while avoiding an obstacle region marked on the road. For inference, we modeled the car system with idealized 4-dimensional unicycle dynamics, where the control inputs are turning velocity and forward acceleration. For each of 7 demonstrations, the most likely constraint was determined by removing constraints violated by the demonstration from the candidate set, then selecting the most likely remaining constraint. This calculation can be performed very quickly - all that is required is to determine which candidate constraints the demonstration violates. As shown in Figure 2, a constraint region near the obstacle location is consistently identified after observing a single human demonstration.

## 5 Conclusion

This paper presented a novel algorithm for maximum likelihood constraint inference that works on MDPs with continuous state spaces. For many continuous dynamical systems, these MDPs provide a much closer approximation to the true underlying dynamics than is possible with tabular spaces. The maximum likelihood framework we adopt works well even for sub-optimal demonstrations, allowing robots to infer constraints on trajectories generated by humans in real-world environments in a self-supervised manner.

## References

[1] Usman Anwar, Shehryar Malik, Alireza Aghasi, and Ali Ahmed. Inverse Constrained Reinforcement Learning. *arXiv:2011.09999 [cs, eess]*, May 2021. URL `http://arxiv.org/abs/2011.09999`. arXiv: 2011.09999.

[2] G. Chou, N. Ozay, and D. Berenson. Learning Constraints From Locally-Optimal Demonstrations Under Cost Function Uncertainty. *IEEE Robotics and Automation Letters*, 5(2):3682–3690, April 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.2974427. Conference Name: IEEE Robotics and Automation Letters.

[3] The garage contributors. Garage: A toolkit for reproducible reinforcement learning research, 2019. URL `https://github.com/rlworkgroup/garage`.

[4] Changshuo Li and Dmitry Berenson. Learning Object Orientation Constraints and Guiding Constraints for Narrow Passages from One Demonstration. In Dana Kulić, Yoshihiko Nakamura, Oussama Khatib, and Gentiane Venture, editors, *2016 International Symposium on Experimental Robotics*, Springer Proceedings in Advanced Robotics, pages 197–210, Cham, 2017. Springer International Publishing. ISBN 978-3-319-50115-4. doi: 10.1007/978-3-319-50115-4_18.

[5] Weiwei Li and Emmanuel Todorov. Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems:. In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pages 222–229, Setúbal, Portugal, 2004. SciTePress - Science and and Technology Publications. ISBN 978-972-8865-12-2. doi: 10.5220/0001143902220229. URL `http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0001143902220229`.

[6] H. Lin, M. Howard, and S. Vijayakumar. Learning null space projections. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2613–2619, May 2015. doi: 10.1109/ICRA.2015.7139551. ISSN: 1050-4729.

[7] C. Karen Liu and Dan Negrut. The Role of Physics-Based Simulators in Robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):35–58, May 2021. ISSN 2573-5144, 2573-5144. doi: 10.1146/annurev-control-072220-093055. URL `https://www.annualreviews.org/doi/10.1146/annurev-control-072220-093055`.

[8] Lucia Pais, Keisuke Umezawa, Yoshihiko Nakamura, and Aude Billard. Learning Robot Skills Through Motion Segmentation and Constraints Extraction. *HRI Workshop on Collaborative Manipulation*, page 5, 2013.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, August 2017. URL `http://arxiv.org/abs/1707.06347`. arXiv: 1707.06347.

[10] Dexter R. R. Scobee and S. Shankar Sastry. Maximum Likelihood Constraint Inference for Inverse Reinforcement Learning. *arXiv:1909.05477 [cs, eess, stat]*, September 2019. URL `http://arxiv.org/abs/1909.05477`. arXiv: 1909.05477.

[11] Kaylene C. Stocking, David L. McPherson, Robert P. Matthew, and Claire J. Tomlin. Discretizing Dynamics for Maximum Likelihood Constraint Inference. *arXiv:2109.04874 [cs, eess]*, September 2021. URL `http://arxiv.org/abs/2109.04874`. arXiv: 2109.04874.

[12] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, page 6, 2008.

## Supplementary Material

### 1.1 Highly Optimal Demonstrators

One drawback to a Monte-Carlo approach to calculating expected constraint violation is that if the demonstrator behaves close to optimally (i.e. with a large $\beta$ parameter), the expected unconstrained trajectories will be tightly clustered around the most optimal path(s). This makes accurate constraint inference difficult due to the paucity of samples with lower rewards, which would only be accrued when acting under a constraint that precludes higher-reward options. We can circumvent this problem by learning a $\pi_\theta$ distribution with a lower $\beta$ than the demonstrator, and then use techniques from importance sampling to correct our results for the true reward function. To formalize this idea, let's say we sample from $\pi_{s,0}(\xi) = \frac{1}{Z_{s,0}} e^{\beta_s R(\xi)}$, but the true reward function induces the distribution $\pi_{d,0}(\xi) = \frac{1}{Z_{d,0}} e^{\beta_d R(\xi)}$. We wish to determine $Z_{d,c_i}/Z_{s,0}$ for each $c_i \in \mathcal{C}$ so that we can identify the likeliest constraint by finding the smallest $Z_{d,c_i}$. Since

$$\frac{Z_{d,c_i}}{Z_{s,0}} = \frac{Z_{d,c_i}}{Z_{s,c_i}} \frac{Z_{s,c_i}}{Z_{s,0}} \tag{5}$$

and we can use Theorem 1 to estimate the second term $Z_{s,c_i}/Z_{s,0}$, all that remains is to calculate the first term $Z_{d,c_i}/Z_{s,c_i}$. The following theorem allows us to do this:

**Theorem 2**

$$\frac{Z_{d,c_i}}{Z_{s,c_i}} = \mathbb{E}_{\xi \sim \pi_{s,c_i}}[e^{(\beta_d - \beta_s)R(\xi)}]$$

(A proof is provided below.) Since we assume deterministic dynamics, sampling from $\pi_{s,c_i}$ can be easily achieved by sampling from $\pi_{s,0}$ and discarding trajectories that violate constraint $c_i$.

### 1.2 Proofs of Theorems

For simplicity, the proofs here assume a continuous state-action space, but similar results hold for the discrete setting. Recall that $q(\xi) = e^{\beta R(\xi)}$. Furthermore, define the normalized trajectory distributions as $\pi_0(\xi) = \frac{1}{Z_0} q(\xi)$ for the nominal unconstrained case and $\pi_c(\xi) = \frac{1}{Z_c} q(\xi) \mathbb{1}_c(\xi)$ for the constrained case. Although it has been omitted for readability, all integrals are over $\xi \in \Xi$.

#### 1.2.1 Proof of Theorem 1:

$$\begin{aligned} \frac{Z_{c_i}}{Z_0} &= \frac{1}{Z_0} \int \mathbb{1}_{c_i}(\xi) d\xi \\ &= \int \pi_0(\xi) \mathbb{1}_{c_i}(\xi) d\xi \\ &= \mathbb{E}_{\xi \sim \pi_0}[\mathbb{1}_{c_i}(\xi)] \end{aligned} \tag{6}$$

Note that the expression for $Z_{c_i}$ comes from removing trajectories that violate $c_i$ from the integral, so that the constrained maximum entropy distribution $\pi_{c_i}$ integrates to 1.

#### 1.2.2 Proof of Theorem 2:

First, define $q_d(\xi) = e^{\beta_d R(\xi)}$, $q_s(\xi) = e^{\beta_s R(\xi)}$, and $\pi_{s,c}(\xi) = \frac{1}{Z_{s,c}} q_s(\xi) \mathbb{1}_c(\xi)$. We have:

$$\begin{aligned} \frac{Z_{d,c}}{Z_{s,c}} &= \frac{1}{Z_{s,c}} \int q_d(\xi) \mathbb{1}_c(\xi) d\xi \\ &= \frac{1}{Z_{s,c}} \int \frac{q_d(\xi)}{\pi_{s,c}(\xi)} \pi_{s,c}(\xi) \mathbb{1}_c(\xi) d\xi \\ &= \frac{1}{Z_{s,c}} \int \frac{e^{\beta_d R(\xi)}}{e^{\beta_s R(\xi)}/Z_{s,c}} \pi_{s,c}(\xi) \mathbb{1}_c(\xi) d\xi \\ &= \int e^{(\beta_d - \beta_s)R(\xi)} \pi_{s,c}(\xi) d\xi \\ &= \mathbb{E}_{\xi \sim \pi_{s,c}}[e^{(\beta_d - \beta_s)R(\xi)}] \end{aligned} \tag{7}$$

The constraint avoidance indicator $\mathbb{1}_c(\xi)$ disappears in line 4 because it is redundant with $\pi_{s,c}(\xi)$.

### 1.3 Experimental design

For both experimental systems, we use proximal policy optimization (PPO) to optimize the objective in equation 4 due to its good performance and stability [9].

#### 1.3.1 Pendulum system

The pendulum model consists of a 2-dimensional state space (angle $\theta$ and angular velocity $\dot{\theta}$). The 1-dimensional control input is the normalized torque applied at the base of the pendulum:

$$\ddot{\theta} = \frac{g}{l} \cdot \sin(\theta) + u \tag{8}$$

The constraint hypothesis set is an evenly spaced grid of 100 non-overlapping cells that cover the state space of $\theta \in [0, 2\pi]$ and $\dot{\theta} \in [-6, 6]$. The demonstrator wants to arrive at a particular goal state $\hat{s_T}$ at the end of a $T$ = 5s period while minimizing the total squared torque and avoiding the true constraint region, $K$. In practice, this is achieved by optimizing the reward function

$$\begin{aligned} u^* = &\arg\max_u \; -[\alpha_1 \|s(T) - \hat{s_T}\|_2^2 + \alpha_2 \int_0^T u(t)^2 dt] \\ &\text{s.t. } s(t) \notin K \quad \forall t \in [0, T] \end{aligned} \tag{9}$$

$\alpha_1 = 80$ and $\alpha_2 = 0.05$ were chosen so that most trajectories terminate in a small neighborhood around the goal state. To optimize this objective for constrained demonstrations, we used the constrained optimal control method described in [11]. Briefly, this method uses 10 iterations of the Iterative Linear-Quadratic Regulator (iLQR) algorithm [5], and outputs the best of 3 random initializations. We generated 100 demonstration trajectories with random start and goal points for each ground-truth constraint $C_1$ and $C_2$. The ground-truth constraints are shown in Figure 1a. After discarding start/goal pairs for which the demonstrator wasn't able to reach the goal state in the 5s time horizon, 65 demonstrations remained for each ground-truth constraint. Note that since the pendulum dynamics are nonlinear, these demonstrations are not guaranteed to be optimal and may correspond to local maxima in the reward function.

The MDP for this system was formulated using a timestep of $\delta t = 1/60s$ and 5 discrete action choices linearly distributed between $u = -1$ and $u = 1$. A separate deep RL policy was trained for each goal state by inserting the reward function in equation 9 into the objective derived in equation 4. Using $\beta = 1.5$ was found to provide the best fit to the demonstrations. Random starting states were used during training. After training, sampling was performed by using the same starting state as the demonstrator and performing 10,000 independent policy rollouts on the simulated pendulum environment. Since each demonstration had a different start and goal state, at inference time we averaged expected constraint violations across the deep RL policies corresponding to each demonstration.

The constraint inference error metric in Figure 1b refers to the rank of the ground truth constraint in the algorithm's ranking of the most likely constraints. The best possible score is 1, which would indicate that the ground truth constraint is always picked as the true constraint. Note that since the demonstrations have random start and end points, some of them will be uninformative about the ground truth constraint, for example in the case where the constrained region is extremely unlikely to be entered by an unconstrained demonstrator.

#### 1.3.2 Model car system

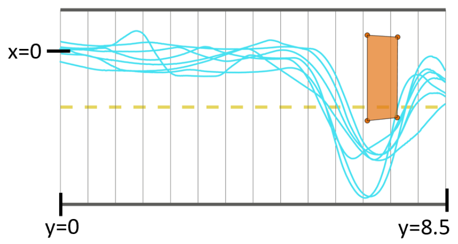

Figure 3: Constrained demonstrations for the car model car system with the true obstacle region (orange) and constraint hypothesis set (empty boxes). This is a version of Figure 2b with the x and y values indicated.

The position of the model car was recorded with an Optitrack system, and the car's speed and heading were inferred from the position data. We modeled the car with idealized unicycle dynamics, which have the following form:

$$\begin{aligned} \dot{x} &= v\cos(\theta) \qquad & \dot{\theta} &= u_1 \\ \dot{y} &= v\sin(\theta) \qquad & \dot{v} &= u_2 \end{aligned} \tag{10}$$

The human attempted to drive to the end of the racetrack segment while avoiding the obstacle region marked on the road and otherwise staying in their lane when possible. We modeled this behavior with the following reward function:

$$\begin{aligned} u^* = \arg\max_u - \int_0^T \left\| \begin{matrix} \alpha_1(x(t) - \hat{x}) \\ \alpha_2(y(t) - \hat{y}) \end{matrix} \right\|_1 dt \\ \text{s.t. } s(t) \notin K \quad \forall t \in [0:T] \end{aligned} \tag{11}$$

where the roadway is aligned with the y-axis so that the car starts at ($x(0)$ = 0, $y(0)$ = 0), $\hat{x} = 0$ is the center of the left lane, and $\hat{y} = 8.5$ is the far end of the road segment. $K$ is an obstacle in the roadway that prevents the human from being in the left lane between $y = 6.7$ and $y = 7.4$, but is unknown to our algorithm. $T$ is a variable time horizon where the episode ends once the car crosses the $y = \hat{y}$ line.

To formulate the MDP model, we choose a discrete action space with an evenly spaced grid of 9 points between ($u_1$ = -0.5, $u_2$ = -1) and ($u_1$ = 0.5, $u_2$ = 1). We use a simulation time-step of $\delta t$ = 0.05s. These values, as well as the parameters $\alpha_1$ = 100 and $\alpha_2$ = 0.05, and the human's temperature parameter $\beta$ = 0.005, were chosen to create a close match between the human's behavior when not attempting to avoid an obstacle (i.e. ignoring the constraint $K$) and a deep RL policy trained with this objective. The constraint hypothesis set is 28 evenly spaced regions along the roadway in $(x, y)$ space, 14 in each lane. The real-world racetrack and a diagram of the corresponding MDP model is shown in Fig. 2. A single deep RL policy was trained to perform the task of driving the car following the reward function in equation 11, with an additional reward bonus for reaching the goal and a penalty for leaving the track. 1000 trajectories were sampled from the resulting $\pi_\theta$ distribution. Trajectories that left the racetrack or did not reach the goal were discarded. The remaining samples allowed us to rank the most likely candidate constraints before observing any demonstrations.