Maximum Likelihood Constraint Inference on Continuous State Spaces

Kaylene C. Stocking[1], D. Livingston McPherson[1], Robert P. Matthew[2], and Claire J. Tomlin[1]

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 demonstrator'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 evaluate our approach on two continuous dynamical systems: a 2-dimensional inverted pendulum model, and a 4-dimensional unicycle model that was successfully used for fast constraint inference on a 1/10 scale car remote-controlled by a human.**

I. 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, assuming the dynamics are known [1]. In many practical situations, however, we already have a good idea of the reward function an agent may be optimizing. Drivers usually want to stay near the center of their lane, and a person walking across a room typically chooses shorter paths over longer ones. In cases like these, even though the reward function is already known, 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 [2]. MLCI uses the maximum entropy framework [1] to identify which constraint in a constraint hypothesis set provides the best explanation for unexpected demonstrator behavior. The constraint hypothesis set can be obtained, for example, by dividing the state space into an even grid. It can also be designed specifically for a particular application, with possible constraints chosen based on their utility or known prevalence. Choosing the hypothesized constraints is analogous to selecting features to be used in the learned reward function in standard inverse reinforcement learning formulations [1]. Although using a constraint hypothesis set limits which constraints it is possible to learn, it also acts as a source of inductive bias and means the results of MLCI tend to be easy to interpret.

Unfortunately, the MLCI algorithm presented in [2] only works for system models with discrete, finite state-action spaces. While it is possible to approximate continuous dynamical systems for use with the algorithm, this process introduces significant error into the dynamics and suffers from an exponential increase in the number of discrete states required as the number of state dimensions in the dynamics increases (often called the "curse of dimensionality") [3]. In this paper, we present a novel MLCI method that works directly with continuous state spaces, allowing for a much closer approximation to continuous real-world dynamics and avoiding the curse of dimensionality in the state-space. Our method first learns an implicit representation of the expected

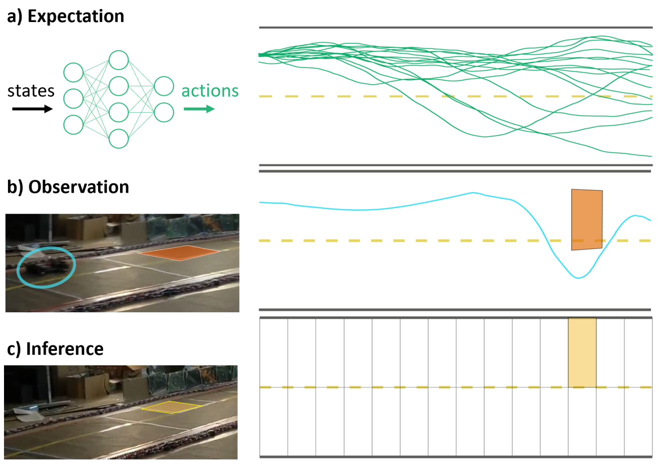

Fig. 1. Our proposed method allows for fast inference of the most likely constraints on demonstrators. Here, we illustrate our approach on a demonstration from a human-controlled model car driving along a racetrack. Given a known reward function and dynamical model, deep reinforcement learning is used to learn an implicit representation of the maximum entropy trajectory distribution, and trajectories are sampled from this distribution (a). These samples provide an estimate of the expected frequency a demonstrator will violate each constraint in a constraint hypothesis set. When an unusual demonstration is observed as the car swerves around an obstacle on the road (b), its constraint violations can be compared with the pre-computed expectations, and our proposed algorithm identifies the constraints that provide the best explanation of the demonstrator's behavior (c).

*This work is supported by ONR under the project "Leveraging egocentric and allocentric representations for navigation", and by the SRC CONIX program. Code used for our experiments is available at `https://github.com/violasox/continuous-state-MLCI`.

[1]Kaylene Stocking, D. Livingston McPherson, and Claire Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California Berkeley, Berkeley, CA 94720, USA `kaylene@berkeley.edu`

[2]Robert Matthew is with the Department of Physical Therapy and Rehabilitation Science, University of California, San Francisco, San Francisco, CA 94143, USA

maximum entropy distribution over demonstrator trajectories by leveraging deep reinforcement learning (RL). Then, we sample from this distribution to obtain Monte Carlo estimates of expected constraint violations, which allows us to perform MLCI. Our method retains several key advantages from [2], including working with sub-optimal demonstrations and allowing for most calculations to be pre-computed before any demonstrations are observed.

The remainder of this paper is laid out as follows. In Section II, we present prior work and discuss the trade-offs between various constraint inference methods. In Section III, we briefly overview the mathematical background for MLCI and describe our algorithm in more detail. Section IV presents analysis on a 2-dimensional pendulum model with demonstrations generated by an optimal control algorithm. Finally, Section V provides an example of successful real-world constraint inference on the trajectory of a 1/10-scale car, driven by a human, that must swerve to avoid an obstacle.

II. Related Work

Different perspectives on constraint inference have yielded a variety of methods which differ in what kinds of systems can be modeled and what assumptions are made about the demonstrator. Our work is most directly inspired by the maximum likelihood method introduced by [2], which identifies the most likely constraint(s) from a hypothesis set even when the demonstrator may behave sub-optimally with respect to their goal. This is achieved by using the maximum entropy framework, which is also leveraged by [4] to learn task specifications. Specifications can be thought of as a generalization of Markovian (memoryless) constraints to complex multi-step behaviors. Both of these methods require discrete state-action spaces, although [3] examines how continuous dynamical systems might be approximated with discrete spaces for the purposes of constraint inference.

Many other methods work directly with continuous dynamics, but drop the maximum likelihood feature. Chou et al. [5] 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. For example, Pais et al. [6] assume that constrained behaviors will have high within-demonstration variance and low between-demonstration variance, and Li et al. [7] uses the heuristic that maintaining a robot end effector in the same orientation throughout a demonstration suggests a constraint. Lin et al. [8] present a kinematics-based approach for learning constraints that affect how a nominal policy is executed in different environments. An approach proposed by Mehr et al. [9] is specialized for online constraint inference in the context of shared autonomy, where mis-identified constraints can be corrected by the user.

Malik et al. [10] recently presented an approach that works with continuous state-action spaces and retains the maximum likelihood framework. They use deep reinforcement learning to identify a constraint function that will allow an agent to avoid violating the constraint(s) affecting the demonstrator. The method we propose similarly uses deep RL to handle continuous state spaces, but focuses instead on learning constraints from a pre-specified constraint hypothesis set. Because of this difference in perspective, our method enables checking for specific possible constraints and leveraging their presence (or absence) to make decisions beyond executing a modified policy. Our method can also leverage prior knowledge of the dynamics and reward function to perform fast online inference. Therefore, our work is likely to be especially useful in situations where the system is known before observing demonstrations.

III. Constraint Inference On Continuous State Spaces

A. Markov Decision Dynamics

Our approach relies on a system model formulated as a deterministic Markov Decision Process (MDP). The MDP is a tuple of four elements:

- A state space $\mathcal{S}$ to navigate and a set of actions $\mathcal{A}$ that can be taken at each state. Both $\mathcal{S}$ and $\mathcal{A}$ may be either continuous or discrete.
- A transition kernel $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ that describes how $a_{t-1} \in \mathcal{A}$ affects s_t. The repeated application of this transition kernel generates a sequence of states over a time horizon $t \in [0 : T]$ given a sequence of action choices. The couple of state sequence and action sequence is a trajectory $\xi = (s_0, a_0, ..., a_{T-1}, s_T)$ and the space of all possible trajectories is Ξ.
- An objective metric $R : \Xi \rightarrow \mathbb{R}$ (also called a reward function) that measures the quality of trajectories.

If the state and action spaces are both discrete and finite, we say that the MDP is tabular. The MLCI approach developed by [2] only works for tabular MDPs. The method we propose in section III-C, however, admits MDPs with discrete or continuous state spaces. For many continuous dynamical systems, this allows for a model that is a much closer approximation to the true dynamics. Note that the MDP formulation requires discrete time steps; however, continuous dynamics can be approximated well with a sufficiently small δt using standard simulation methods [11].

B. Maximum Entropy Likelihood on Trajectories

After observing a demonstrator's behavior, we suspect that they may be avoiding a constraint we don't know about when their trajectory is surprising, in the sense of accumulating significantly less reward than should be possible. We can formalize this intuition by defining a distribution of expected demonstrator behavior. Following [1], we adopt the maximum entropy likelihood distribution, under which the probability of a trajectory on a tabular MDP is defined as:

$$P(\xi) = \frac{1}{Z} e^{\beta R(\xi)} = \frac{1}{Z} q(\xi) \tag{1}$$

$$Z = \sum_{\xi' \in \Xi} e^{\beta R(\xi')} \tag{2}$$

where β is a temperature parameter that reflects how noisy (sub-optimal) the demonstrator is, $q(\xi)$ is the unnormalized trajectory probability, and Z is a normalizing constant. For continuous state-action spaces, this distribution can be generalized to a probability density:

$$p(\xi) = \frac{e^{\beta R(\xi)}}{\int_{\xi' \in \Xi} e^{\beta R(\xi')} d\xi'} = \frac{1}{Z} q(\xi) \tag{3}$$

To remain agnostic as to whether the state-action space is continuous or discrete, we define the maximum entropy distribution as $\pi(\xi) = \frac{1}{Z} q(\xi)$ for both cases.

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 c, we obtain a new maximum entropy distribution:

$$\pi_c(\xi) = \begin{cases} \frac{1}{Z_c} q(\xi), \ (s_t, a_t) \notin c \ \forall t \in [0:T] \\ 0, \text{otherwise} \end{cases} \tag{4}$$

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 new constraints, 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 ξ 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 appendix.)

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 under this distribution. For tabular MDPs, both components can be obtained via dynamic programming [2]. Although a method for approximating continuous dynamical systems with tabular MDPs was presented in [3], it is desirable to work directly with continuous state spaces and avoid the approximation errors inherent to the discretization process. This motivates a new approach.

C. A Sampling-based Method for Continuous MLCI

In this section we describe our algorithm for constraint inference on MDPs with continuous state spaces.

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 the policy gradient method in reinforcement learning (RL). Following [10], consider the KL-divergence between the true maximum entropy distribution $\pi_0(\xi)$ and a learned policy $\pi_\theta(\xi)$ parameterized by θ.

$$\begin{aligned} D_{KL}(\pi_\theta || \pi_0) &= \mathbb{E}_{\xi \sim \pi_\theta}[\log\pi_\theta(\xi) - \log\pi_0(\xi)] \\ &= \mathbb{E}_{\xi \sim \pi_\theta}[\log\pi_\theta(\xi) - \beta R(\xi) + \log Z_0] \end{aligned} \tag{5}$$

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

$$J(\pi_\theta) = \mathbb{E}_{\xi \sim \pi_\theta}[R(\xi) - \frac{1}{\beta}\log\pi_\theta(\xi)] \tag{6}$$

This objective can be used with any policy gradient method. In our experiments, we use proximal policy optimization (PPO) due to its good performance and stability [12].

The parameterization of π_θ 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 [13]). 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 handle continuous action spaces by discretizing the range of possible action inputs. A categorical action policy can then be used to learn action distributions with an arbitrary shape at each state.

2) Sampling-Based Approximation of Z_{c_i}/Z_0: The learned policy π_θ 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 be very close to the true constraint violation probability (the probability that the absolute error δ is larger than ϵ decays exponentially with the number of samples n: $P(\delta > \epsilon) \leq 2e^{-2\epsilon^2 n}$). By Theorem 1, calculating the fraction of trajectories that violate each hypothesized constraint is sufficient for determining the most likely constraint. If constraint regions are determined by actions (for example, a constraint on the amount of torque that can be applied with a motor), extra care may need to be taken with determining how the actions are discretized and constraint violation is determined for sampled trajectories. In this work we focus on state-based constraints.

3) Real-Time Constraint Inference: Once we have learned the policy π_θ and calculated expected constraint violation via Monte Carlo sampling, we are ready to perform constraint inference. This last step simply consists of sorting by magnitude the estimated Z_{c_i}/Z_0 for each hypothesized constraint, then checking this ranking against the constraints violated by the demonstrations. The smallest Z_{c_i}/Z_0 that is not violated by any demonstration is the most likely constraint, as described in section III-B. It is important to note that prior to this final step of checking which constraints are violated by the demonstrations, no information from the demonstrations is used by our algorithm. Therefore, even though the policy learning and sampling steps described in this section may be time-consuming, they can be pre-computed when the MDP and constraint hypothesis set are known in advance. This means that constraint inference can occur in real-time as demonstrations are collected, a key difference from prior work that requires demonstrations to be available throughout the deep learning process [10].

4) 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 β 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 trajectory samples with lower rewards under the unconstrained distribution, since these lower-reward trajectories would only occur when acting under a constraint. We can circumvent this problem by learning a π_θ distribution with a lower β than the demonstrator, and then using techniques from importance sampling to correct our results for the true reward function. To formalize this idea, consider the case where 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 $\pi_{d,0}(\xi) = \frac{1}{Z_{d,0}} e^{\beta_d R(\xi)}$. If we can determine $Z_{d,c_i}/Z_{s,0}$ for each $c_i \in \mathcal{C}$, this allows us to determine which Z_{d,c_i} is smallest and therefore identify the likeliest constraint. Since

$$\frac{Z_{d,c_i}}{Z_{s,0}} = \frac{Z_{d,c_i}}{Z_{s,c_i}} \frac{Z_{s,c_i}}{Z_{s,0}} \tag{7}$$

and we can use Theorem 1 to estimate $Z_{s,c_i}/Z_{s,0}$, all that remains is to calculate $Z_{d,c_i}/Z_{s,c_i}$. 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 in the appendix.) Since we assume deterministic dynamics, sampling from π_{s,c_i} can be easily achieved by sampling from $\pi_{s,0}$ and discarding trajectories that violate constraint c_i.

IV. Analysis with Pendulum System

A. Pendulum Dynamics

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

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

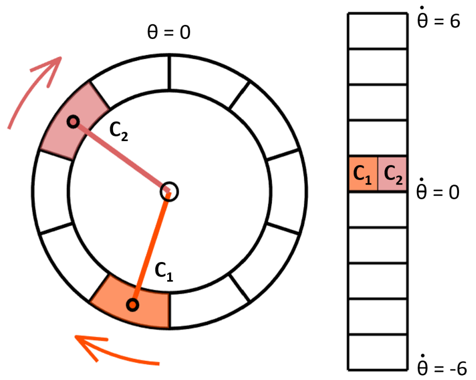

Fig. 2. This figure illustrates the ground truth constraints for our pendulum dynamics analysis. The constraint hypothesis space evenly divides the state space into 100 cells, 10 along the angle axis and 10 along angular velocity. The two constraints used in our experiments, C_1 and C_2, are shown here in different shades of orange. The constraints cover different angle regions but the same angular velocity.

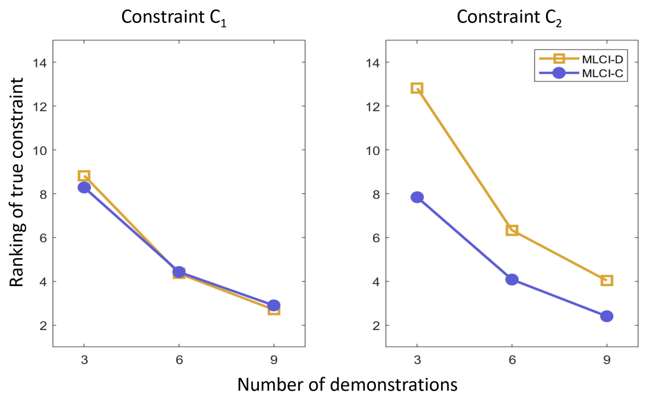

Fig. 3. Constraint inference accuracy for our method (MLCI-Continuous, blue circles), with the performance of the discretized state space method from [3] (MLCI-Discrete, gold squares) as a comparison. The y-axis measures the likelihood ranking of the ground truth constraint among all 100 hypothesized constraints, on average. A score of 1 means that the true constraint is identified as the most likely.

Where the gravitational constant g and the length of the pendulum l are both assumed to be 1 for simplicity. The constraint hypothesis set is an evenly spaced 10-by-10 grid of non-overlapping cells that cover the state space of $\theta \in [0, 2\pi]$ and $\dot{\theta} \in [-6, 6]$, for a total of 100 possible constraints. We assign a uniform prior likelihood to every constraint. 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

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

using the constrained optimal control method described in [3]. $\alpha_1 = 80$ and $\alpha_2 = 0.05$ were chosen so that most trajectories terminate in a small neighborhood around the goal state. Note that since the pendulum dynamics are nonlinear, the demonstrations generated with this method are not guaranteed to be globally optimal and may correspond to local maxima in the reward function.

B. Experimental Design

We examined two possible ground truth constraints, shown in Fig. 2. We generated 65 demonstration trajectories with random start and goal points for each constraint. 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 of the 65 goal states by inserting the reward function in equation 9 into the objective derived in equation 6. Using $\beta = 0.67$ 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.

C. Constraint Inference Results

To analyze constraint inference performance on the pendulum system, we pick demonstrations at random (across 200 trials) and use the average constraint violations from the corresponding trajectory samples to rank the most likely constraints. We compare the performance of our algorithm to the best hyperparameter settings for the approximate discrete state space approach proposed in [3]. (A direct comparison to the work of Malik et al. [10], which also performs MLCI on continuous systems, is not possible since this method does not utilize a constraint hypothesis set.) The results for each method are shown in Fig. 3. Our approach 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 discrete state space method for C_1, which lies in a region of the pendulum state space that is close to a stable equilibrium point. However, our method performs comparatively better at identifying 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.

V. Constraint Inference on 1/10 Scale Car

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 Traxxas robotic car along a racetrack, attempting to stay in their lane and reach the goal $(\hat{x}, \hat{y})$ quickly (reward) while avoiding an obstacle region marked on the road (constraint). The position of the car was recorded with an Optitrack system, and the car's speed and heading were inferred from the position data.

A. MDP formulation

We modeled the car system with idealized 4-dimensional unicycle dynamics, where the control inputs are turning velocity (u_1) and forward acceleration (u_2). These dynamics 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 is assumed to be optimizing the following reward function:

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

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 right 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 right 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 δt = 0.05s. These values, as well as the reward function parameters α_1 = 100 and α_2 = 0.05, and the human's temperature parameter β = 200, 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 the unconstrained reward function. 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 are shown in Fig. 4. A 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 π_θ 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.

B. Constraint inference

7 demonstration trajectories were collected. For each demonstration, the most likely constraint after observing the single demonstration was determined by removing constraints violated by the demonstration from the candidate set, then selecting the most likely remaining constraint using the pre-computed $Z_{s,c_i}/Z_{s,0}$ estimates. This calculation can be performed very quickly - all that is required is to determine which candidate constraints the demonstration violates. As shown in the bottom-right panel of Fig. 4, for 3 of the demonstrations, the candidate constraint that most closely matches the true constraint region is identified as the most likely constraint. For the remaining 4 demonstrations, a neighboring candidate constraint is selected, although it is still a close approximation to the ground truth constraint. This second region is selected because the human turns

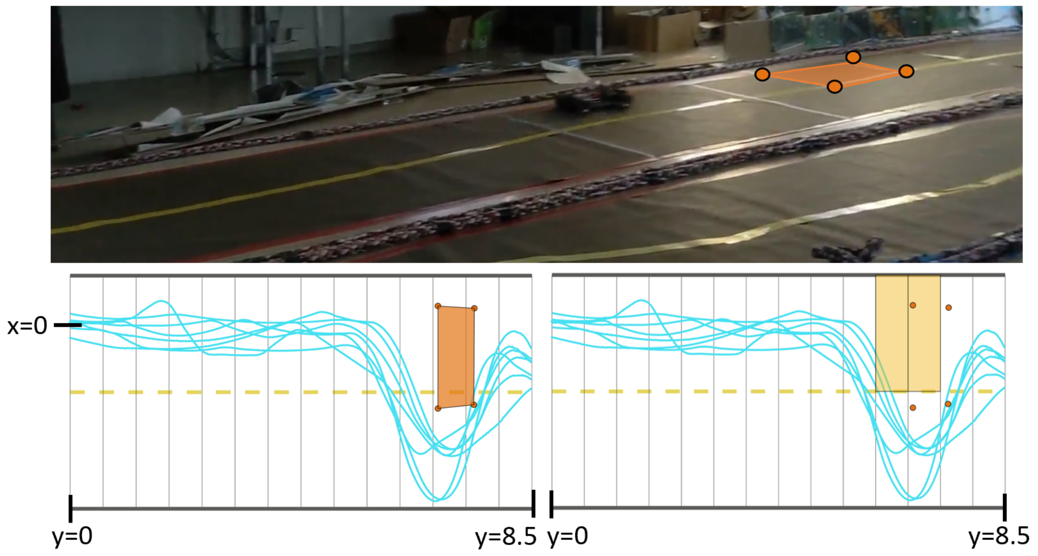

Fig. 4. Constraint inference on a 1/10-scale car trajectory along a racetrack. The upper panel shows a picture of the car approaching the obstacle region, highlighted in orange. The bottom panels show the x and y components of the corresponding MDP space with 7 human demonstration trajectories plotted. The grid in these panels shows the hypothesized constraint set, where each constraint is a region in (x,y) space. In the bottom left panel, the ground truth constraint is highlighted. In the bottom right panel, the possible constraints identified as most likely after observing a single demonstration are shown. The left-hand constraint is the most likely for 3 demonstrations, while the neighboring region to the right is the most likely for the remaining 4 demonstrations. The orange circles mark the position of the true constraint, which is close to the inferred constraints.

slightly earlier than is necessary to avoid the obstacle in these demonstrations.

VI. 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, which suffer from the curse of dimensionality. Our algorithm works with continous state spaces by leveraging a Monte Carlo sampling approach to track expected constraint violation, which requires only an implicit representation of the maximum entropy trajectory distribution. For continuous state spaces, deep reinforcement learning yields such a representation. Although our method does rely on a discrete action space and discrete transition dynamics, we show in our experiments that accurate constraint inference is nevertheless possible for interesting continuous systems. By inferring constraints from a user-specified hypothesis set, we obtain an interpretable result (although this does come at the cost of restricting which constraints can be inferred). Our approach also allows for significant pre-computation, so that only a trivial filtering process is required at the time demonstrations are observed. The maximum likelihood framework we use is well-suited to noisy or sub-optimal demonstrators including non-expert humans, enables real-time constraint inference, and leads to interpretable and actionable inferences.

APPENDIX

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 $p_0(\xi) = \frac{1}{Z_0}q(\xi)$ for the nominal unconstrained case and $p_c(\xi) = \frac{1}{Z_c}q(\xi)\mathbb{1}_c(\xi)$ for the constrained case.

Proof of Theorem 1:

$$\begin{aligned}\frac{Z_{c_i}}{Z_0} &= \frac{1}{Z_0}\int q(\xi)\mathbb{1}_{c_i}(\xi)d\xi \\ &= \int p_0(\xi)\mathbb{1}_{c_i}(\xi)d\xi \\ &= \mathbb{E}_{\xi\sim p_0}[\mathbb{1}_{c_i}(\xi)]\end{aligned} \tag{12}$$

Note that the expression for Z_{c_i} comes from removing trajectories that violate c_i from the integral, so that the constrained probability distribution p_{c_i} integrates to 1.

Proof of Theorem 2:

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

$$\begin{aligned}\frac{Z_{d,c}}{Z_{s,c}} &= \frac{1}{Z_{s,c}}\int q_d(\xi)\mathbb{1}_c(\xi)d\xi \\ &= \frac{1}{Z_{s,c}}\int \frac{q_d(\xi)}{p_{s,c}(\xi)}p_{s,c}(\xi)\mathbb{1}_c(\xi)d\xi \\ &= \frac{1}{Z_{s,c}}\int \frac{e^{\beta_d R(\xi)}}{e^{\beta_s R(\xi)}/Z_{s,c}}p_{s,c}(\xi)\mathbb{1}_c(\xi)d\xi \\ &= \int e^{(\beta_d-\beta_s)R(\xi)}p_{s,c}(\xi)d\xi \\ &= \mathbb{E}_{\xi\sim p_{s,c}}[e^{(\beta_d-\beta_s)R(\xi)}]\end{aligned} \tag{13}$$

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

References

[1] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, p. 6, 2008.

[2] D. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *International Conference on Learning Representations*, 2019.

[3] K. C. Stocking, D. L. McPherson, R. P. Matthew, and C. J. Tomlin, "Discretizing dynamics for maximum likelihood constraint inference," Sep. 10, 2021. arXiv: `2109.04874`. [Online]. Available: `http://arxiv.org/abs/2109.04874`.

[4] M. Vazquez-Chanlatte, S. Jha, A. Tiwari, M. K. Ho, and S. Seshia, "Learning task specifications from demonstrations," *Advances in Neural Information Processing Systems*, vol. 31, pp. 5367–5377, 2018. [Online]. Available: `https://proceedings.neurips.cc/paper/2018/hash/74934548253bcab8490ebd74afed7031-Abstract.html` (visited on 11/09/2020).

[5] G. Chou, N. Ozay, and D. Berenson, "Learning constraints from locally-optimal demonstrations under cost function uncertainty," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3682–3690, Apr. 2020, Conference Name: IEEE Robotics and Automation Letters, ISSN: 2377-3766. DOI: `10.1109/LRA.2020.2974427`.

[6] L. Pais, K. Umezawa, Y. Nakamura, and A. Billard, "Learning robot skills through motion segmentation and constraints extraction," *HRI Workshop on Collaborative Manipulation*, p. 5, 2013.

[7] C. Li and D. Berenson, "Learning object orientation constraints and guiding constraints for narrow passages from one demonstration," in *2016 International Symposium on Experimental Robotics*, D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, Eds., ser. Springer Proceedings in Advanced Robotics, Cham: Springer International Publishing, 2017, pp. 197–210, ISBN: 978-3-319-50115-4. DOI: `10.1007/978-3-319-50115-4_18`.

[8] H. Lin, M. Howard, and S. Vijayakumar, "Learning null space projections," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, ISSN: 1050-4729, May 2015, pp. 2613–2619. DOI: `10.1109/ICRA.2015.7139551`.

[9] N. Mehr, R. Horowitz, and A. D. Dragan, "Inferring and assisting with constraints in shared autonomy," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 6689–6696. DOI: `10.1109/CDC.2016.7799299`.

[10] S. Malik, U. Anwar, A. Aghasi, and A. Ahmed, "Inverse constrained reinforcement learning," in *International Conference on Machine Learning*, PMLR, 2021, pp. 7390–7399.

[11] C. K. Liu and D. Negrut, "The role of physics-based simulators in robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 35–58, May 3, 2021, ISSN: 2573-5144, 2573-5144. [Online]. Available: `https://www.annualreviews.org/doi/10.1146/annurev-control-072220-093055`.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347 [cs]*, Aug. 28, 2017. arXiv: `1707.06347`. [Online]. Available: `http://arxiv.org/abs/1707.06347`.

[13] The garage contributors, *Garage: A toolkit for reproducible reinforcement learning research*, 2019. [Online]. Available: `https://github.com/rlworkgroup/garage`.