Learning Constraints from Demonstrations with Grid and Parametric Representations

Journal Title
XX(X):1–??

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Glen Chou, Dmitry Berenson, and Necmiye Ozay[1]

Abstract

We extend the learning from demonstration paradigm by providing a method for learning unknown constraints shared across tasks, using demonstrations of the tasks, their cost functions, and knowledge of the system dynamics and control constraints. Given safe demonstrations, our method uses hit-and-run sampling to obtain lower cost, and thus unsafe, trajectories. Both safe and unsafe trajectories are used to obtain a consistent representation of the unsafe set via solving an integer program. Our method generalizes across system dynamics and learns a guaranteed subset of the constraint. Additionally, by leveraging a known parameterization of the constraint, we modify our method to learn parametric constraints in high dimensions. We also provide theoretical analysis on what subset of the constraint and safe set can be learnable from safe demonstrations. We demonstrate our method on linear and nonlinear system dynamics, show that it can be modified to work with suboptimal demonstrations, and that it can also be used to learn constraints in a feature space.

Keywords

learning from demonstration, machine learning, motion and path planning

1 Introduction

Inverse optimal control and inverse reinforcement learning (IOC/IRL) (Ratliff et al. (2006); Abbeel and Ng (2004); Argall et al. (2009); Ng and Russell (2000)) have proven to be powerful tools in enabling robots to perform complex goal-directed tasks. These methods learn a cost function that replicates the behavior of an expert demonstrator when optimized. However, planning for many robotics and automation tasks also requires knowing constraints, which define what states or trajectories are safe. For example, the task of safely and efficiently navigating an autonomous vehicle can naturally be described by a cost function trading off user comfort and efficiency and by the constraints of collision avoidance and executing only legal driving behaviors. In some situations, constraints can provide a more interpretable representation of a behavior than cost functions. For example, in safety critical environments, recovering a hard constraint or an explicit representation of an unsafe set in the environment is more useful than learning a "softened" cost function representation of the constraint as a penalty term in the Lagrangian. Consider the autonomous vehicle, which absolutely must avoid collision, not simply give collisions a cost penalty. Furthermore, learning global constraints shared across many tasks can be useful for generalization. Again consider the autonomous vehicle, which must avoid the scene of a car accident: this is a shared constraint that holds regardless of the task it is trying to complete.

While constraints are important, it can be impractical for a user to exhaustively program into a robot all the possible constraints that it should obey when performing its repertoire of tasks. To avoid this, we consider in this paper the problem of recovering the latent constraints within expert demonstrations that are shared across tasks in the environment. Our method is based on the key insight that each safe, optimal demonstration induces a set of lower-cost trajectories that must be unsafe due to violation of an unknown constraint. Our method samples these unsafe trajectories, ensuring they are also consistent with the known constraints (system dynamics, control constraints, and start/goal constraints), and uses these unsafe trajectories together with the safe demonstrations as constraints in an "inverse" integer program which recovers a consistent unsafe set. Our contributions are fivefold:

- We pose the novel problem of learning a shared constraint across tasks.
- We propose an algorithm that, given known constraints and boundedly suboptimal demonstrations of state-control sequences, extracts unknown constraints defined in a wide range of constraint spaces (not limited to the trajectory or state spaces) shared across demonstrations of different tasks.
- We propose a variant of the aforementioned algorithm which can scale more gracefully to constraints in high dimensions by assuming and leveraging parametric structure in the constraint.
- We provide theoretical analysis on the limits of what subsets of a constraint can be learned, depending

[1]Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 48109, USA.

Corresponding author:
Glen Chou, University of Michigan, 1301 Beal Avenue, Ann Arbor, 48109.
Email: gchou@umich.edu

on the demonstrations, the system dynamics, and the trajectory discretization. We also prove that our method can recover guaranteed inner approximations of both the unsafe set and the safe set.

- We provide experiments that justify our theory and show that our algorithm can recover an unsafe set with few demonstrations, across different types of linear and nonlinear dynamics, and can be adapted to work with boundedly suboptimal demonstrations. We also demonstrate that our method can learn constraints in high-dimensional state spaces and parameter spaces.

2 Related Work

Inverse optimal control (Kalman (1964); Keshavarz et al. (2011)) (IOC) and **inverse reinforcement learning** (IRL) (Ng and Russell (2000)) aim to recover an objective function consistent with the received expert demonstrations, in the sense that the demonstrations (approximately) optimize the cost function. Our method is complementary to these approaches; if the demonstration is solving a constrained optimization problem, we are finding its constraints, given the objective function; IOC/IRL finds the objective function, given its constraints. For example, Englert et al. (2017) attempts to learn the cost function of a constrained optimization problem from optimal demonstrations by minimizing the residuals of the KKT conditions, but the constraints themselves are assumed known. On the other hand, a risk-sensitive approach to IRL is proposed in Singh et al. (2018) and is complementary to our work, which aims to learn hard constraints. Another approach in Amin et al. (2017) can represent a state-space constraint shared across tasks as a penalty term in the reward function of an MDP. However, when viewing a constraint as a penalty, it becomes unclear if a demonstrated motion was performed to avoid a penalty or to improve the cost of the trajectory in terms of the true cost function (or both). Thus, learning a constraint which generalizes between tasks with different cost functions becomes difficult. To avoid this issue, we assume a known cost function to explicitly reason about the constraint.

One branch of **safe reinforcement learning** aims to perform exploration while minimizing visitation of unsafe states. Several methods for safe exploration in the state space (Schreiter et al. (2015); Turchetta et al. (2016); Akametalu et al. (2014)) use a Gaussian process (GP) to explore safe regions in the state space. These approaches differ from ours in that they use exploration instead of demonstrations. Some drawbacks to these methods include that unsafe states can still be visited, Lipschitz continuity of the safety function is assumed, or the dynamics are unknown but the safe set is known. Furthermore, states themselves are required to be explicitly labeled as safe or unsafe, while we only require that entire trajectories be labeled as safe. Our method is capable of learning a binary constraint defined in other spaces, using only state-control trajectories.

There exists prior work in learning **geometric constraints** in the workspace. In Armesto et al. (2017), a method is proposed for learning Pfaffian constraints, recovering a linear constraint parametrization. In Pérez-D'Arpino and Shah (2017), a method is proposed to learn geometric constraints which can be described by the classes of considered constraint templates. Our method generalizes these methods by being able to learn a nonlinear constraint defined in any constraint space (not limited to the state space).

Learning **local trajectory-based constraints** has also been explored in the literature. The method in Li and Berenson (2016) samples feasible poses around waypoints of a single demonstration; areas where few feasible poses can be sampled are assumed to be constrained. Similarly, Mehr et al. (2016) performs online constraint inference in the feature space from a single trajectory, and then learns a mapping to the task space. The methods in Pais et al. (2013); Ye and Alterovitz (2011); Calinon and Billard (2008, 2007) also learn constraints in a single task. These methods are inherently local since only one trajectory or task is provided, unlike our method, which aims to learn a global constraint shared across tasks.

Our work is also relevant to **human-robot interaction**. In Knepper et al. (2017), implicit communication of facts between agents is modeled as an interplay between demonstration and inference, where "surprising" demonstrations trigger inference of the implied fact. Our method can be seen as an inference algorithm which infers an unknown constraint implied by a "surprising" demonstration.

This paper builds upon the constraint learning algorithms presented in Chou et al. (2018) and Chou et al. (2019). The algorithm presented in Chou et al. (2018) relies on a discretization of the state space, yielding a constraint recovery method which scales exponentially with the constraint space dimension. The method in Chou et al. (2019) extends Chou et al. (2018) by modifying the formulation of the constraint recovery problem, leveraging a known parameterization of the constraint to enable inference of safe and unsafe sets in high-dimensional constraint spaces. Furthermore, several new examples demonstrate the effectiveness of the new method in high-dimensional settings. This journal paper presents a unified view of the methods presented in Chou et al. (2018) and Chou et al. (2019), with a detailed discussion comparing the benefits and shortcomings of the two approaches. We also present a more detailed theoretical analysis of the aforementioned algorithms and derive tightened theoretical guarantees for the algorithm in Chou et al. (2018). Additionally, we compare our approach with inverse reinforcement learning, demonstrating that learning a constraint as a cost penalty can lead to unsafe behavior.

3 Preliminaries and Problem Statement

The goal of this work is to recover unknown constraints shared across a collection of optimization problems, given boundedly suboptimal solutions, the cost functions, and knowledge of the dynamics, control constraints, and start/goal constraints. We discuss the forward problem, which generates the demonstrations, and the inverse problem: the core of this work, which recovers the constraints.

3.1 Forward optimal control problem

Consider an agent described by a state in some state space $x \in \mathcal{X}$. It can take control actions $u \in \mathcal{U}$ to change its state. The agent performs tasks Π drawn from a set of

tasks $\mathcal{P}$, where each task Π can be written as a constrained optimization problem over state trajectories ξ_x in state trajectory space $\mathcal{T}^x$ and control trajectories ξ_u in control trajectory space $\mathcal{T}^u$:

Problem 1. Forward problem / "task" Π.

$$\begin{aligned} \underset{\xi_x,\xi_u}{\text{minimize}} \quad & c_\Pi(\xi_x,\xi_u) \\ \text{s.t.} \quad & \phi(\xi_x,\xi_u) \in \mathcal{S} \subseteq \mathcal{C} \\ & \bar{\phi}(\xi_x,\xi_u) \in \bar{\mathcal{S}} \subseteq \bar{\mathcal{C}} \\ & \phi_\Pi(\xi_x,\xi_u) \in \mathcal{S}_\Pi \subseteq \mathcal{C}_\Pi \end{aligned} \tag{1}$$

where $c_\Pi(\cdot) : \mathcal{T}^x \times \mathcal{T}^u \to \mathbb{R}$ is a cost function for task Π and $\phi(\cdot,\cdot) : \mathcal{T}^x \times \mathcal{T}^u \to \mathcal{C}$ is a known feature function mapping state-control trajectories to some constraint space $\mathcal{C}$, elements of which are referred to as "constraint states". Mappings $\bar{\phi}(\cdot,\cdot) : \mathcal{T}^x \times \mathcal{T}^u \to \bar{\mathcal{C}}$ and $\phi_\Pi(\cdot,\cdot) : \mathcal{T}^x \times \mathcal{T}^u \to \mathcal{C}_\Pi$ are known and map to potentially different constraint spaces $\bar{\mathcal{C}}$ and $\mathcal{C}_\Pi$, containing a known shared safe set $\bar{\mathcal{S}}$ and a known task-dependent safe set $\mathcal{S}_\Pi$, respectively. $\mathcal{S}$ is an unknown safe set, and the inverse problem aims to recover its complement, $\mathcal{A} \doteq \mathcal{S}^{c*}$, the "unsafe" set. In this paper, we focus on constraints separable in time: $\phi(\xi_x,\xi_u) \in \mathcal{A} \Leftrightarrow \exists t \in \{1,\dots,T\}\ \phi(\xi_x(t),\xi_u(t)) \in \mathcal{A}$, where we overload ϕ so it applies to the instantaneous values of the state and the input. An analogous definition holds for the continuous time case. Our method can also learn constraints which are partially or completely inseparable in time (i.e. $\phi(\xi_x,\xi_u) \in \mathcal{A} \Leftrightarrow \exists \{t_i,\dots,t_j\} \in \{1,\dots,T\}\ \phi(\xi_x(t),\xi_u(t)) \in \mathcal{A}, \forall t \in \{t_i,\dots,t_j\}$)[†].

A demonstration, $\xi_{xu} \doteq (\xi_x,\xi_u) \in \mathcal{T}^{xu}$, is a state-control trajectory which is a boundedly suboptimal solution to Problem 1, i.e. the demonstration satisfies all constraints and its cost is at most a factor of δ above the cost of the optimal solution ξ^*_{xu}, i.e. $c(\xi^*_x,\xi^*_u) \le c(\xi_x,\xi_u) \le (1+\delta)c(\xi^*_x,\xi^*_u)$. Furthermore, let T be a finite time horizon which is allowed to vary. If ξ_{xu} is a discrete-time trajectory ($\xi_x = \{x_1,\dots,x_T\}$, $\xi_u = \{u_1,\dots,u_T\}$), Problem 1 is a finite-dimensional optimization problem, while Problem 1 becomes a functional optimization problem if ξ_{xu} is a continuous-time trajectory ($\xi_x : [0,T] \to \mathcal{X}$, $\xi_u : [0,T] \to \mathcal{U}$). We emphasize that this setup does not restrict the unknown constraint to be defined on the trajectory space; it allows for constraints to be defined on any space described by the range of some known feature function ϕ.

We assume the trajectories are generated by a dynamical system $\dot{x} = f(x,u,t)$ or $x_{t+1} = f(x_t,u_t,t)$ with control constraints $u_t \in \mathcal{U}$, for all t, and that the dynamics, control constraints, and start/goal constraints are known. We further denote the set of state-control trajectories satisfying the unknown shared constraint, the known shared constraint, and the known task-dependent constraint as $\mathcal{T}_\mathcal{S}$, $\mathcal{T}_{\bar{\mathcal{S}}}$, and $\mathcal{T}_{\mathcal{S}_\Pi}$, respectively. Lastly, we also denote the set of trajectories satisfying all known constraints but violating the unknown constraint as $\mathcal{T}_\mathcal{A}$.

3.2 Inverse constraint learning problem

The goal of the inverse constraint learning problem is to recover an unsafe set, $\mathcal{A} \subseteq \mathcal{C}$, using N_s provided safe demonstrations $\xi^*_{s_j}, j = 1,\dots,N_s$, known constraints, and $N_{\neg s}$ inferred unsafe trajectories, $\xi_{\neg s_k}, k = 1,\dots,N_{\neg s}$, generated by our method, which can come from multiple tasks. The safe and unsafe trajectories can together be thought of as a set of constraints on the possible assignments of unsafe constraint states in $\mathcal{C}$.

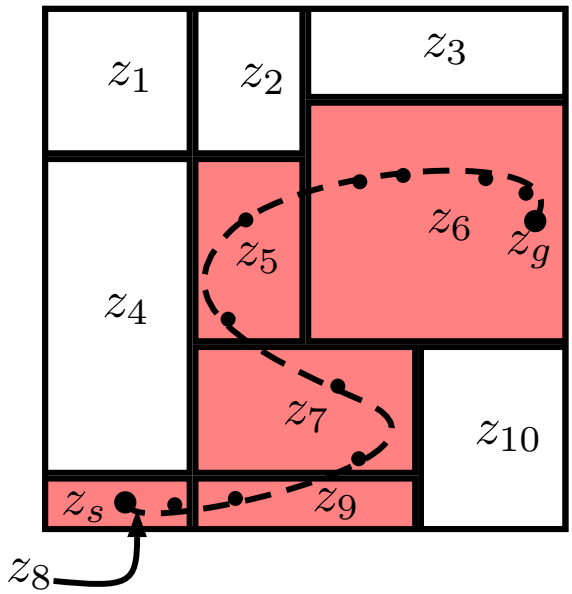

Figure 1. *Discretized constraint space with cells $z_1,\dots,z_{10}$. The trajectory's constraint values are assigned to the red cells.*

Depending on the amount of structure that we assume we know about the constraint, there are two approaches. If no structure about the constraint is known at all, the constraint space can be gridded and unsafeness can be learned on a grid-by-grid basis (Section 3.2.1). Otherwise, if the constraint is known to be described by some parameterization, the parameters can be learned, which leads to better scalability (Section 3.2.2).

Inferring unsafe trajectories, i.e. obtaining $\xi_{\neg s_k}, k = 1,\dots,N_{\neg s}$, is the most difficult part of this problem, since finding lower-cost trajectories consistent with known constraints that complete a task is essentially a planning problem. Much of Section 4 shows how to efficiently obtain $\xi_{\neg s_k}$.

3.2.1 Recovering a gridded constraint To recover a gridded approximation of the unsafe set $\mathcal{A}$ that is consistent with these trajectories, we first discretize $\mathcal{C}$ into a finite set of G discrete cells $\mathcal{Z} \doteq \{z_1,\dots,z_G\}$ and define an occupancy function, $\mathcal{O}(\cdot)$, which maps each cell to its safeness: $\mathcal{O}(\cdot) : \mathcal{Z} \to \{0,1\}$, where $\mathcal{O}(z_i) = 1$ if $z_i \in \mathcal{A}$, and 0 otherwise[‡]. Continuous space trajectories are gridded by concatenating the set of grid cells z_i that $\phi(x_1),\dots,\phi(x_T)$ lie in. With slight abuse of notation, we will use $z_i \in \phi(\xi_{s_i})$ to denote

*To be exact, the safe set is assumed to be compact for the forward problem (Problem 1) to be well-defined, and we aim to learn the closure of the unsafe set $\mathcal{A} \doteq \text{cl}(\mathcal{S}^c)$.

[†]This can be done by writing the constraints of Problem 2 as sums over partially separable/inseparable feature components instead of completely separable components.

[‡]To avoid complications when states lie on the boundary shared between two grid cells, grid cells are defined to be disjoint open sets.

$z_i \in \{\phi(\xi_{s_i}(1)), \ldots, \phi(\xi_{s_i}(T_i))$. The grid discretization is graphically shown in Figure 1 with a non-uniform grid. Then, the problem can be written down as an integer feasibility problem:

Problem 2. Inverse feasibility problem.

$$\begin{aligned} \text{find} \quad & \mathcal{O}(z_1), \ldots, \mathcal{O}(z_G) \in \{0,1\}^G \\ \text{s.t.} \quad & \sum_{z_i \in \phi(\xi^*_{s_j})} \mathcal{O}(z_i) = 0, \quad \forall j = 1, \ldots, N_s \\ & \sum_{z_i \in \phi(\xi_{\neg s_k})} \mathcal{O}(z_i) \geq 1, \quad \forall k = 1, \ldots, N_{\neg s} \end{aligned} \tag{2}$$

Further details on Problem 2, including conservativeness guarantees, incorporating a prior on the constraint, and a continuous relaxation is presented in Section 4.4.

3.2.2 Recovering a parametric constraint Suppose that the unsafe set can be described by some parameterization $\mathcal{A}(\theta) \doteq \{k \in \mathcal{C} \mid g(k, \theta) \leq 0\}$, where $g(\cdot, \cdot)$ is known and θ are parameters to be learned. Then, a feasibility problem analogous to Problem 2 can be written to find a feasible θ consistent with the data:

Problem 3. Parametric constraint recovery problem.

$$\begin{aligned} \text{find} \quad & \theta \\ \text{s.t.} \quad & g(k, \theta) > 0, \quad \forall k \in \phi(\xi^*_{s_i}), \quad \forall i = 1, \ldots, N_s \\ & \big(\exists k \in \phi(\xi_{\neg s_j}), \quad g(k, \theta) \leq 0\big), \quad \forall j = 1, \ldots, N_{\neg s} \end{aligned}$$

Further details on Problem 3, including conservativeness guarantees and specific mixed-integer programming formulations for common constraint parameterizations are presented in Section 4.5.

4 Method

The key to our method lies in finding lower-cost trajectories that do not violate the known constraints, given a demonstration with boundedly-suboptimal cost satisfying all constraints. Such trajectories must then violate the unknown constraint, and we extend existing sampling algorithms to be more efficient for trajectory-space sampling under various assumptions on the dynamics. Our goal is to determine an unsafe set in the constraint space from these trajectories using either Problem 2 or Problem 3. In the following, Section 4.1 describes lower-cost trajectories consistent with the known constraints; Section 4.2 describes how to sample such trajectories; Section 4.3 describes how to get more information from unsafe trajectories; Section 4.4 describes details and extensions to Problem 2; Section 4.5 describes details and extensions to Problem 3; Section 4.6 discusses how to extend our method to suboptimal demonstrations. The complete flow of our method is described in Algorithm 2.

4.1 Trajectories satisfying known constraints

Consider the forward problem (Problem 1). We define the set of unsafe state-control trajectories induced by an optimal, safe demonstration ξ^*_{xu}, $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$, as the set of state-control trajectories of lower cost that obey the known constraints:

$$\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}} \doteq \{\xi_{xu} \in \mathcal{T}_{\bar{\mathcal{S}}} \cap \mathcal{T}_{\mathcal{S}_\Pi} \mid c(\xi_x, \xi_u) < c(\xi^*_x, \xi^*_u)\}. \tag{3}$$

In this paper, we deal with the known constraints from the system dynamics, the control limits, and task-dependent start and goal state constraints. Hence, $\mathcal{T}_{\bar{\mathcal{S}}} = \mathcal{D}^{\xi_{xu}} \cap \mathcal{U}^{\xi_{xu}}$, where $\mathcal{D}^{\xi_{xu}}$ denotes the set of dynamically feasible trajectories and $\mathcal{U}^{\xi_{xu}}$ denotes the set of trajectories using controls in $\mathcal{U}$ at each time-step. $\mathcal{T}_{\mathcal{S}_\Pi}$ denotes trajectories satisfying start and goal constraints. We develop the method for discrete time trajectories, but analogous definitions hold in continuous time. For discrete time, length T trajectories, $\mathcal{U}^{\xi_{xu}}$, $\mathcal{D}^{\xi_{xu}}$, and $\mathcal{T}_{\mathcal{S}_\Pi}$ are the following subsets of $\mathcal{T}^{xu}$:

$$\begin{aligned} \mathcal{U}^{\xi_{xu}} &\doteq \{\xi_{xu} \mid u_t \in \mathcal{U}, \ \forall t \in \{1, \ldots, T-1\}\}, \\ \mathcal{D}^{\xi_{xu}} &\doteq \{\xi_{xu} \mid x_{t+1} = f(x_t, u_t), \ \forall t \in \{1, \ldots, T-1\}\}, \\ \mathcal{T}_{\mathcal{S}_\Pi} &\doteq \{\xi_{xu} \mid x_1 = x_s, x_T = x_g\}. \end{aligned} \tag{4}$$

4.2 Sampling trajectories satisfying known constraints

We sample from $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ to obtain lower-cost trajectories obeying the known constraints using hit-and-run sampling (Kiatsupaibul et al. (2011)), a method guaranteeing convergence to a uniform distribution of samples over $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ in the limit; the method is detailed in Algorithm 1 and an illustration is shown in Figure 2. Hit-and-run starts from an initial point within the set, chooses a direction uniformly at random, moves a random amount in that direction such that the new point remains within the set, and repeats.

Depending on the convexity of the cost function and the control constraints and on the form of the dynamics, different sampling techniques can be used, organized in Table 1. The following sections describe each sampling method.

Algorithm 1: Hit-and-run

Output: out $\doteq \{\xi_1, \ldots, \xi_{N_{\neg s}}\}$
Input : $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}, \xi^*_{xu}, N_{\neg s}$
$\xi_{xu} \leftarrow \xi^*_{xu}$; out $\leftarrow \{\}$;
for i = $1{:}N_{\neg s}$ **do**
  $r \leftarrow$ sampleRandDirection();
  $\mathcal{L} \leftarrow \mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}} \cap \{\xi'_{xu} \in \mathcal{T} \mid \xi'_{xu} = \xi_{xu} + \beta r\}$;
  $L_-, L_+ \leftarrow$ endpoints($\mathcal{L}$);
  $\xi_{xu} \sim$ Uniform(L_-, L_+);
  out $\leftarrow$ out $\cup\, \xi_{xu}$;
end

4.2.1 Ellipsoid hit-and-run When we have a linear system with quadratic cost and convex control constraints, a very common setup in the optimal control literature, the set of lower-cost trajectories satisfying the known constraints

$$\begin{aligned} \mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}} &\doteq \{\xi_{xu} \mid c(\xi_{xu}) < c(\xi^*_{xu})\} \cap \mathcal{D}^{\xi_{xu}} \\ &\equiv \{\xi_{xu} \mid \xi_{xu}^\top V \xi_{xu} < \xi^{*\top}_{xu} V \xi^*_{xu}\} \cap \mathcal{D}^{\xi_{xu}} \end{aligned}$$

is an ellipsoid in the trajectory space, which can be efficiently sampled via a specially-tailored hit-and-run method. Here, the quadratic cost is written as $c(\xi_{xu}) \doteq \xi_{xu}^\top V \xi_{xu}$, where V is a matrix of cost parameters, and we omit the control and task constraints for now. Consider the intersection of the random line chosen from hit-and-run (r in Algorithm

Dynamics	Cost function	Control constraints	Sampling method
Linear	Quadratic	Convex	Ellipsoid hit-and-run (Section 4.2.1)
Linear	Convex	Convex	Convex hit-and-run (Section 4.2.2)
Else			Non-convex hit-and-run (Section 4.2.3)

Table 1. Sampling methods for different classes of dynamics models, cost functions, and feasible control sets.

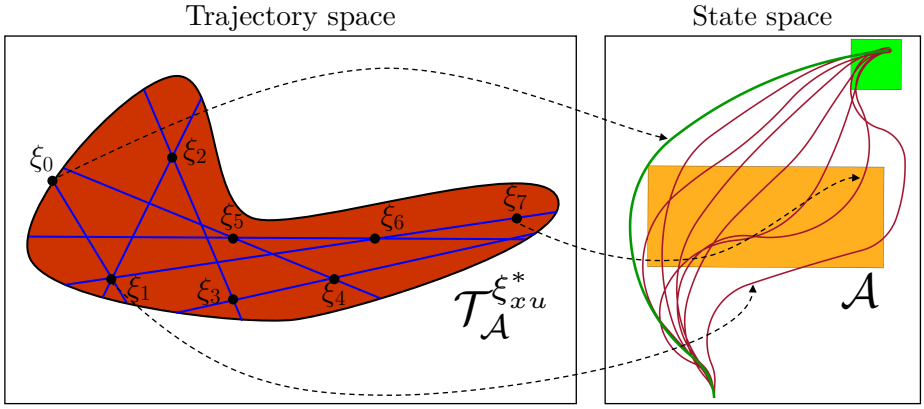

Figure 2. *Illustration of hit-and-run.* ***Left****: Blue lines denote sampled random directions, black dots denote samples.* ***Right****: Each point in $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ corresponds to an unsafe trajectory in the constraint space $\mathcal{C}$, and in this case, $\mathcal{C} = \mathcal{X}$.*

1) with the lower-cost trajectory set $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$; denote this line segment as $\mathcal{L}$ and its endpoints as L_- and L_+ (c.f. Algorithm 1). Furthermore, denote β as a step-size in direction r; hence, L_- and L_+ put bounds on the allowable step-sizes β. Without dynamics, L_-, L_+ can be found by solving a quadratic equation $L_-^\top V L_+ = (\xi_{xu} + \beta_1 r)^\top V (\xi_{xu} + \beta_2 r) = \xi_{xu}^{*\top} V \xi^*_{xu}$. We show that this can still be done with linear dynamics by writing $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ in a special way. $\mathcal{D}^{\xi_{xu}}$ can be written as an eigenspace of a singular "dynamics consistency" matrix, D_1, which converts any arbitrary state-control trajectory to one that satisfies the dynamics, one time-step at a time. Precisely, if the dynamics can be written as $x_{t+1} = Ax_t + Bu_t$, we can write a matrix D_1:

$$\underbrace{\begin{bmatrix} x_1 \\ u_1 \\ x_2 \\ u_2 \\ \tilde{x}_3 \\ \vdots \\ u_{T-1} \\ \tilde{x}_T \end{bmatrix}}_{\hat{\xi}_{xu}} = \underbrace{\begin{bmatrix} I & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & I & 0 & 0 & 0 & \cdots & \cdots & 0 \\ A & B & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 0 & I & 0 & \cdots & \cdots & 0 \\ 0 & 0 & A & B & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 0 & I & 0 \\ 0 & 0 & 0 & \cdots & \cdots & A & B & 0 \end{bmatrix}}_{D_1} \underbrace{\begin{bmatrix} x_1 \\ u_1 \\ \tilde{x}_2 \\ u_2 \\ \tilde{x}_3 \\ \vdots \\ u_{T-1} \\ \tilde{x}_T \end{bmatrix}}_{\xi_{xu}} \tag{5}$$

that fixes the controls and the initial state and performs a one-step rollout, replacing the second state with the dynamically correct state. In (5), we denote by $\tilde{x}_{t+1}$ a state that cannot be reached by applying control u_t to state x_t. Multiplying the one-step corrected trajectory $\hat{\xi}_{xu}$ by D_1 again changes $\tilde{x}_3$ to the dynamically reachable state x_3. Applying D_1 to the original T-time-step infeasible trajectory $T-1$ times results in a dynamically feasible trajectory, $\xi_{xu}^{\text{feas}} = D_1^{T-1}\xi_{xu}$. Further, note that the set of dynamically feasible trajectories is $\mathcal{D}^{\xi_{xu}} \doteq \{\xi_{xu} \mid D_1 \xi_{xu} = \xi_{xu}\}$, which is the span of the eigenvectors of D_1 associated with eigenvalue 1. Thus, obtaining a feasible trajectory via repeated multiplication is akin to finding the eigenspace via power iteration (Golub and Van Loan (1996)). One can also interpret this as propagating through the dynamics with a fixed control sequence. Now, we can write $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ as another ellipsoid:

$$\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}} \doteq \{\xi_{xu} \mid \xi_{xu}^\top {D_1^{T-1}}^\top V D_1^{T-1} \xi_{xu} \leq \xi_{xu}^{*\top} V \xi^*_{xu}\}. \tag{6}$$

Like for the kinematic case, this ellipsoid can be efficiently sampled after finding L_-, L_+ by solving a quadratic equation $(\xi_{xu} + \beta_1 r)^\top {D_1^{T-1}}^\top V D_1^{T-1} (\xi_{xu} + \beta_2 r) = \xi_{xu}^{*\top} V \xi^*_{xu}$.

We deal with control constraints separately, as the intersection of $\mathcal{U}^{\xi_{xu}}$ and (6) is in general not an ellipsoid. To ensure control constraint satisfaction, we reject samples with controls outside of $\mathcal{U}^{\xi_{xu}}$; this works if $\mathcal{U}^{\xi_{xu}}$ is not measure zero. For task constraints, we ensure all sampled rollouts obey the goal constraints by adding a large penalty term to the cost function: $\tilde{c}(\cdot) \doteq c(\cdot) + \alpha_c \|x_g - x_T\|_2^2$, where α_c is a large scalar, which can be incorporated into (6) by modifying V and including x_g in ξ_{xu}; all trajectories sampled in this modified set satisfy the goal constraints to an arbitrarily small tolerance ε, depending on the value of α_c. The start constraint is satisfied trivially: all rollouts start at x_s. Note the demonstration cost remains the same, since the demonstration satisfies the start and goal constraints; this modification is made purely to ensure these constraints hold for sampled trajectories.

4.2.2 Convex hit-and-run For general convex cost functions, the same sampling method holds, but L_+, L_- cannot be found by solving a quadratic function. Instead, we solve $c(\xi_{xu} + \beta r) = c(\xi^*_{xu})$ via a root finding algorithm or line search.

4.2.3 Non-convex hit-and-run If $\mathcal{T}_{\mathcal{A}}^{\xi^*_{xu}}$ is non-convex, $\mathcal{L}$ can now in general be a union of disjoint line segments. In this scenario, we perform a "backtracking" line search by setting β to lie in some initial range: $\beta \in [\underline{\beta}, \overline{\beta}]$; sampling

β_s within this range and then evaluating the cost function to see whether or not $\xi_{xu} + \beta_s r$ lies within the intersection. If it does, the sample is kept and hit-and-run proceeds normally; if not, then the range of possible β values is restricted to $[\beta_s, \overline{\beta}]$ if β_s is negative, and $[\underline{\beta}, \beta_s]$ otherwise. Then, new βs are re-sampled until either the interval length shrinks below a threshold or a feasible sample is found. This altered hit-and-run technique still converges to a uniform distribution on the set in the limit, but has a slower mixing time than for the convex case, where mixing time describes the number of samples needed until the total variation distance to the steady state distribution is less than a small threshold (Abbasi-Yadkori et al. (2017)). Further, we accelerate sampling spread by relaxing the goal constraint to a larger tolerance $\hat{\varepsilon} > \varepsilon$ but keeping only the trajectories reaching within ε of the goal.

Algorithm 2: Overall method

Output: $\mathcal{O} \doteq \mathcal{O}(z_1), \ldots, \mathcal{O}(z_G)$ (Problems 2, 4, 5)
θ (Problems 3, 6)
Input : $\xi_s = \{\xi_1^*, \ldots, \xi_{N_s}^*\}$, $c_\Pi(\cdot)$, known constraints}

$\xi_u \leftarrow \{\}$;
for *i = 1:N_s* **do**
  /* Sample unsafe ξ */
  if *lin., quad., conv.* **then**
    $\xi_u \leftarrow \xi_u \cap$ ellipsoidHNR(ξ_i^*);
  else if *lin., conv., conv.* **then**
    $\xi_u \leftarrow \xi_u \cap$ convexHNR(ξ_i^*);
  else
    $\xi_u \leftarrow \xi_u \cap$ nonconvexHNR(ξ_i^*);
end
/* Constraint recovery */
if *gridded* **then**
  $\mathcal{O} \leftarrow$ Problem X(ξ_s, ξ_u);
  /* X = 5 if prior, continuous */
  /* X = 4 if prior, binary */
  /* X = 2 if no prior */
else if *parameterization* **then**
  $\theta \leftarrow$ Problem Y(ξ_s, ξ_u);
  /* Y = 6 if polytope param. */
  /* Y = 3 if general param. */

4.3 Improving learnability using cost function structure

Naïvely, the sampled unsafe trajectories may provide little information. Consider an unsafe, length-T discrete-time trajectory ξ, with start and end states in the safe set. This only says there exists at least one intermediate unsafe state in the trajectory, but says nothing directly about which state was unsafe. The weakness of this information can be made concrete using the notion of a version space. In machine learning, the version space is the set of consistent hypotheses given a set of examples (Russell and Norvig (2003)). In our setting, hypotheses are possible unsafe sets, and examples are the safe and unsafe trajectories. Knowing ξ is unsafe only disallows unsafe sets that mark every constraint state in the constraint space that ξ traverses as safe: $(\mathcal{O}(z_2) = 0) \wedge \ldots \wedge (\mathcal{O}(z_{T-1}) = 0)$. If $\mathcal{C}$ is gridded into G cells, this information invalidates at most 2^{G-T+2} out of 2^G possible unsafe sets. We could do exponentially better if we reduced the number of cells that ξ implies could be unsafe.

We can achieve this by sampling sub-segments (or sub-trajectories) of the larger demonstrations, holding other portions of the demonstration fixed. For example, say we fix all but one of the points on ξ when sampling unsafe lower-cost trajectories. Since only one state can be different from the known safe demonstration, the unsafeness of the trajectory can be uniquely localized to whatever new point was sampled: then, this trajectory will reduce the version space by at most a factor of 2, invalidating at most $2^G - 2^{G-1} = 2^{G-1}$ unsafe sets. One can sample these sub-trajectories in the full-length trajectory space by fixing appropriate waypoints during sampling: this ensures the full trajectory has lower cost and only perturbs desired waypoints. However, to speed up sampling, sub-trajectories can be sampled directly in the lower dimensional sub-trajectory space if the cost function $c(\cdot)$ that is being optimized is strictly monotone (Morin (1982)): for any costs $c_1, c_2 \in \mathbb{R}$, control $u \in \mathcal{U}$, and state $x \in \mathcal{X}$, $c_1 < c_2 \Rightarrow h(c_1, x, u) < h(c_2, x, u)$, for all x, u, where $h(c, x, u)$ represents the cost of starting with initial cost c at state x and taking control u. Strictly monotone cost functions include separable cost functions with additive or multiplicative stage costs, which are common in motion planning and optimal control. If the cost function is strictly monotone, we can sample lower-cost trajectories from sub-segments of the optimal path; otherwise it is possible that even if a new sub-segment with lower cost than the original sub-segment were sampled, the full trajectory containing the sub-segment could have a higher cost than the demonstration.

4.4 Gridded integer program formulation

As mentioned in Sections 3.2.1 and 3.2.2, we can solve various optimization problems after sampling to find an unsafe set consistent with the safe and unsafe trajectories. We now discuss the details of this process, starting with the gridded formulation (Problem 2).

4.4.1 Conservative estimate One can obtain a conservative estimate of the unsafe set $\mathcal{A}$ from Problem 2 by intersecting all possible solutions: if the unsafeness of a cell is shared across all feasible solutions, that cell must be occupied. In practice, it may be difficult to directly find all solutions to the feasibility problem, as in the worst case, finding the set of all feasible solutions is equivalent to exhaustive search in the full gridded space (Papadimitriou and Steiglitz (1982)). A more efficient method is to loop over all G grid cells $z_1, \ldots, z_G$ and set each one to be safe, and see if the optimizer can still find a feasible solution. Cells where there exists no feasible solution are guaranteed unsafe. This amounts to solving G binary integer feasibility problems, which can be trivially parallelized. Furthermore, any cells that are known safe (from demonstrations) do not need to be checked. We use this method to compute the "*learned guaranteed unsafe cells*", $\mathcal{G}_{\neg_s}^z$, in Section 6, which we define as:

$$\mathcal{G}_{\neg_s}^z = \{z \in \{z_1, \ldots, z_G\} \mid \mathcal{O}(z) = 1,\ \forall \mathcal{O} \in \mathcal{F}^z\} \quad (7)$$

where $\mathcal{F}^z$ is the feasible set of Problem 2.

4.4.2 A prior on the constraint As will be further discussed in Section 5.1, it may be fundamentally impossible to recover a unique unsafe set. If we have some prior on the nature of the unsafe set, such as it being simply connected, or that certain regions of the constraint space are unlikely to be unsafe, we can make the constraint learning problem more well-posed. Assume that this prior knowledge can be encoded in some "energy" function $E(\cdot,\dots,\cdot):\{0,1\}^G \to \mathbb{R}$ mapping the set of binary occupancies to a scalar value, which indicates the desirability of a particular unsafe set configuration. Using E as the objective function in Problem 2 results in a binary integer program, which finds an unsafe set consistent with the safe and unsafe trajectories, and minimizes the energy:

Problem 4. Inverse binary minimization constraint recovery.

$$\begin{aligned}\underset{\substack{\mathcal{O}(z_1),\dots,\mathcal{O}(z_G)\\ \in\{0,1\}^G}}{\text{minimize}} \quad & E(\mathcal{O}(z_1),\dots,\mathcal{O}(z_G))\\ \text{s.t.} \quad & \sum_{z_i\in\phi(\xi^*_{s_j})} \mathcal{O}(z_i)=0, \quad \forall j=1,\dots,N_s\\ & \sum_{z_i\in\phi(\xi_{\neg s_k})} \mathcal{O}(z_i)\geq 1, \quad \forall k=1,\dots,N_{\neg s}\end{aligned} \tag{8}$$

4.4.3 Probabilistic setting and continuous relaxation A similar problem can be posed in a probabilistic setting, where grid cell occupancies represent beliefs over unsafeness: instead of the occupancy of a cell being an indicator variable, it is instead a random variable Z_i, where Z_i takes value 1 with probability $\tilde{\mathcal{O}}(Z_i)$ and value 0 with probability $1-\tilde{\mathcal{O}}(Z_i)$. Here, the occupancy probability function maps cells to occupancy probabilities $\tilde{\mathcal{O}}(\cdot):\mathcal{Z}\to[0,1]$.

Trajectories can now be unsafe with some probability. We obtain analogous constraints from the integer program in Section 4.4 in the probabilistic setting. Known safe trajectories traverse cells that are unsafe with probability 0; we enforce this with the constraint $\sum_{Z_i\in\phi(\xi^*_{s_j})}\tilde{\mathcal{O}}(Z_i)=0$: if the unsafeness probabilities are all zero along a trajectory, then the trajectory must be safe. Trajectories that are unsafe with probability p_k satisfy $\sum_{Z_i\in\phi(\xi_{\neg s_k})}\tilde{\mathcal{O}}(Z_i)=\mathbb{E}[\sum_{Z_i\in\phi(\xi_{\neg s_k})}Z_i]=(1-p_k)\cdot 0+p_k\cdot S_k\geq p_k$ where we denote the number of unsafe grid cells $\phi(\xi_{\neg s_k})$ traverses when the trajectory is unsafe as S_k, where $S_k\geq 1$. The following problem directly optimizes over occupancy probabilities:

Problem 5. Inverse continuous minimization constraint recovery.

$$\begin{aligned}\underset{\substack{\mathcal{O}(Z_1),\dots,\mathcal{O}(Z_G)\\ \in[0,1]^G}}{\text{minimize}} \quad & E(\mathcal{O}(Z_1),\dots,\mathcal{O}(Z_G))\\ \text{s.t.} \quad & \sum_{z_i\in\phi(\xi^*_{s_j})} \tilde{\mathcal{O}}(Z_i)=0, \quad \forall j=1,\dots,N_s\\ & \sum_{z_i\in\phi(\xi_{\neg s_k})} \tilde{\mathcal{O}}(Z_i)\geq p_k, \quad \forall k=1,\dots,N_{\neg s}\end{aligned} \tag{9}$$

When $p_k=1$, for all k (i.e. all unsafe trajectories are unsafe for sure), this probabilistic formulation coincides with the continuous relaxation of Problem 4. This justifies interpreting the solution of the continuous relaxation as occupancy probabilities for each cell. Note that Problem 4 and 5 have no conservativeness guarantees and use prior assumptions to make the problem more well-posed. However, we observe that they improve constraint recovery in our experiments.

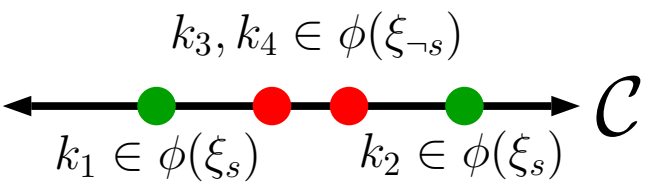

Figure 3. *Given an interval parameterization of an unsafe set, there does not exist any interval which can both explain the data and label and constraint state left of k_1 or right of k_2 as unsafe.*

4.5 Parameter space integer program

Having discussed extensions to the gridded constraint recovery problem, we now turn to analogous results for the parametric case.

4.5.1 Conservative estimate Denote by $\mathcal{F}$ the feasible set of Problem 3. Denote by $\mathcal{G}_{\neg s}$ and $\mathcal{G}_s$ the set of constraint states which are learned guaranteed unsafe and safe, respectively; that is, a constraint state $k\in\mathcal{G}_{\neg s}$ or $k\in\mathcal{G}_s$ if k is classified unsafe or safe for all $\theta\in\mathcal{F}$:

$$\mathcal{G}_{\neg s}\doteq\bigcap_{\theta\in\mathcal{F}}\{k\mid g(k,\theta)\leq 0\} \tag{10}$$

$$\mathcal{G}_{s}\doteq\bigcap_{\theta\in\mathcal{F}}\{k\mid g(k,\theta)> 0\} \tag{11}$$

In contrast to $\mathcal{G}^z_{\neg s}$, which is the set of guaranteed learned unsafe grid cells (the analogue of $\mathcal{G}_{\neg s}$ for grid cells), $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$ are defined directly over the constraint space $\mathcal{C}$.

Note that unlike Problem 2, for Problem 3, it is possible to learn that a constraint state not lying on a demonstration is guaranteed safe. This is due to the parameterization: given a particular set of safe and unsafe trajectories, there may not be any feasible parameter $\theta\in\mathcal{F}$ where k is classified unsafe. For example, consider the case in Figure 3: given the interval parameterization $g(k,\theta=[\bar{k},\underline{k}])=(\bar{k}-k)(\underline{k}-k)$, it is not possible for any constraint state left of k_1 or right of k_2 to be classified unsafe and be consistent with the data. On the other hand, due to the independence of the grids in Problem 2, learning that a given grid cell is safe or unsafe cannot ever imply that another grid cell is guaranteed safe.

Similarly to Problem 2, one can check if a constraint state $k\in\mathcal{G}_s$ or $k\in\mathcal{G}_{\neg s}$ by adding a constraint $g(k,\theta)\leq 0$ or $g(k,\theta)>0$ to Problem 3 and checking feasibility of the resulting program; if the program is infeasible, $k\in\mathcal{G}_s$ or $k\in\mathcal{G}_{\neg s}$. In other words, solving this modified integer program can be seen as querying an oracle about the safety of a constraint state k. The oracle can then return that k is guaranteed safe (program infeasible after forcing k to be unsafe), guaranteed unsafe (program infeasible after forcing k to be safe), or unsure (program remains feasible despite forcing k to be safe or unsafe).

Since Problem 3 works in the continuous constraint space, it is not possible to exhaustively check if each constraint

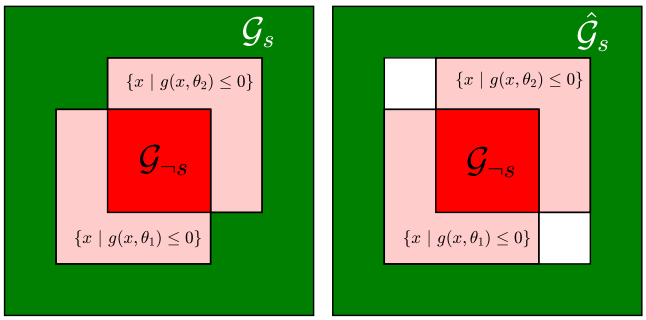

Figure 4. *Comparison of the true $\mathcal{G}_s$ (left, in green) and the extracted inner approximation $\hat{\mathcal{G}}_s$ (right, in green).*

state is guaranteed learned safe or unsafe, unlike the discrete gridded case in Problem 2. For general parameterizations, only individual states can be checked for membership in $\mathcal{G}_s$ or $\mathcal{G}_{\neg s}$. However, for some particularly common parameterizations, there are more efficient methods for recovering subsets of $\mathcal{G}_s$ and $\mathcal{G}_{\neg s}$:

- Axis-aligned hyper-rectangle parameterization: $\mathcal{C} \subseteq \mathbb{R}^n$, $\theta = [\underline{k}_1, \bar{k}_1, \ldots, \underline{k}_n, \bar{k}_n]$, $g(k,\theta) \leq 0 \Leftrightarrow H(\theta) \leq h(\theta)$, where $H(\theta)k = [I_{n\times n}, -I_{n\times n}]^\top$ and $h(\theta) = [\bar{k}_1, \ldots, \bar{k}_n, \underline{k}_1, \ldots, \underline{k}_n]^\top$. Here, $\underline{k}_i$ and $\bar{k}_i$ are the lower and upper bounds of the hyper-rectangle for coordinate i.

  As the set of axis-aligned hyper-rectangles is closed under intersection, $\mathcal{G}_{\neg s}$ is also an axis-aligned hyper-rectangle, the axis-aligned bounding box of any two constraint states $k_1, k_2 \in \mathcal{G}_{\neg s}$ is also contained in $\mathcal{G}_{\neg s}$. This also implies that $\mathcal{G}_{\neg s}$ can be fully described by finding the top and bottom corners $[\underline{k}_1, \ldots, \underline{k}_n]^\top$ and $[\bar{k}_1, \ldots, \bar{k}_n]^\top$. Suppose we start with a known $k \in \mathcal{G}_{\neg s}$. Then, finding $[\underline{k}_1, \ldots, \underline{k}_n]^\top$ amounts to performing a binary search for each of the n dimensions, and the same holds for finding $[\bar{k}_1, \ldots, \bar{k}_n]^\top$.

  Recovering $\mathcal{G}_s$ is not as straightforward, as the complement of axis-aligned boxes is not closed under intersection. However, an inner approximation of $\mathcal{G}_s$ can be efficiently obtained as follows: starting at a constraint state $k \in \mathcal{G}_{\neg s}$, $2n$ line searches can be performed to find the two points of transition to $\mathcal{G}_{\neg s}$ in each constraint coordinate. Denote as $\hat{\mathcal{G}}_s$ the complement of the axis-aligned bounding box of these $2n$ points; $\hat{\mathcal{G}}_s$ is an inner approximation of $\mathcal{G}_s$, as $\mathcal{G}_s = (\bigcap_{\theta\in\mathcal{F}}\{x \mid g(x,\theta) \leq 0\})^c \supseteq \text{AABB}(\bigcap_{\theta\in\mathcal{F}}\{x \mid g(x,\theta) \leq 0\})^c$, where $\text{AABB}(\cdot)$ denotes the axis-aligned bounding box of a set of points and the complement acts on the axis-aligned bounding box.

  For example, consider the scenario in Figure 4 where there are only two feasible parameters, θ_1 and θ_2. Here, $\mathcal{G}_s$ is $(\mathcal{A}(\theta_1) \cup \mathcal{A}(\theta_2))^c$ and $\hat{\mathcal{G}}_s$ under-approximates the safe set ($\mathcal{G}_s$ is in general not representable as the complement of an axis-aligned box).

- Convex parameterization: for fixed θ, $\{k \mid g(k,\theta) \leq 0\}$ is convex.

  While in this case, it is not easy to recover $\mathcal{G}_{\neg s}$ exactly, a subset of $\mathcal{G}_{\neg s}$ can be extracted efficiently by taking the convex hull of any $k_1, k_2, \ldots \in \mathcal{G}_{\neg s}$, since the convex hull is the minimal convex set containing $k_1, k_2, \ldots$.

The same approaches apply for recovering $\mathcal{G}_s$ when it is instead the safe set which is an axis-aligned hyper-rectangle or a convex set.

4.5.2 Choice of parameterization In this section, we identify classes of parameterizations for which Problem 3 can be efficiently solved:

- $g(k,\theta)$ is defined by a Boolean conjunction of linear inequalities, i.e. $\mathcal{A}(\theta)$ can be defined as the union and intersection of half-spaces:

  For this case, mixed-integer programming can be employed. As an example for the particular case where $g(k,\theta) \leq 0$ is a single polytope, i.e. $g(k,\theta) \Leftrightarrow H(\theta)k \leq h(\theta)$, where $H(\theta)$ and $h(\theta)$ are affine in θ, the following mixed integer feasibility problem can be solved to find a feasible θ:

  Problem 6. Polytopic constraint recovery problem.

$$\begin{aligned}
\text{find} \quad & \theta, \{b_s^i\}_{i=1}^{N_s}, \{b_{\neg s}^i\}_{i=1}^{N_{\neg s}} \\
\text{s.t.} \quad & H(\theta)k_i > h(\theta) - M(1-b_s^i), \quad b_{s_j}^i \in \{0,1\}^{N_h}, \\
& \sum_{i=1}^{N_h} b_{s_j}^i \geq 1, \forall k_i \in \phi(\xi_{s_j}), i = 1,\ldots,T_j, j = 1,\ldots,N_s \qquad \text{(12a)} \\
& H(\theta)k_i \leq h(\theta) + M(1-b_{\neg s_k}^i)\mathbf{1}_{N_h}, \quad b_{\neg s_k}^i \in \{0,1\}, \\
& \sum_{i=1}^{T_j} b_{\neg s_k}^i \geq 1, \quad \forall k_i \in \phi(\xi_{\neg s_k}), \quad \forall k = 1,\ldots,N_{\neg s} \qquad \text{(12b)}
\end{aligned}$$

  where M is a large positive number and $\mathbf{1}_{N_h}$ is a column vector of ones of length N_h. Constraints (12a) and (12b) use the big-M formulation to enforce that each safe constraint state lies outside $\mathcal{A}(\theta)$ and that at least one constraint state on each unsafe trajectory lies inside $\mathcal{A}(\theta)$.

  Similar problems can be written down when the safe or unsafe set can be described by unions of polytopes.

  As an alternative to mixed integer programming, satisfiability modulo theories (SMT) solvers can also be employed to solve Problem 3 if $g(k,\theta)$ is defined by a Boolean conjunction of linear inequalities.

- $g(k,\theta)$ is defined by a Boolean conjunction of convex inequalities, i.e. $\mathcal{A}(\theta)$ can be described as the union and intersection of convex sets:

  For this case, satisfiability modulo convex optimization (SMC) (Shoukry et al. (2018)) can be employed to find a feasible θ.

4.5.3 Remarks on parameter space problem We close this subsection with some remarks on implementation and extensions to Problem 3.

- For suboptimal demonstrations or imperfect lower-cost trajectory sampling, Problem 6 can become infeasible. To address this, slack variables can be introduced: replace constraint $\sum_{i=1}^{T_j} b^i_{\neg s} \geq s_k, s_k \in \{0,1\}$ and change the feasibility problem to minimization of $\sum_{k=1}^{N_{\neg s}}(1-s_k)$.
- In addition to recovering sets of guaranteed learned unsafe and safe constraint states, a probability distribution over possibly unsafe constraint states can be estimated by sampling unsafe sets from the feasible set of Problem 3.

4.6 Bounded suboptimality of demonstrations

If we are given a δ-suboptimal demonstration $\hat{\xi}$, where $c(\xi^*) \leq c(\hat{\xi}) \leq (1+\delta)c(\xi^*)$, where ξ^* is an optimal demonstration, we can still apply the sampling techniques discussed in earlier sections, but we must ensure that sampled unsafe trajectories are truly unsafe: a sampled trajectory ξ' of cost $c(\xi') \geq c(\xi^*)$ can be potentially safe. Two options follow: one is to only keep trajectories with cost less than $\frac{c(\hat{\xi})}{1+\delta}$, but this can cause little to be learned if δ is large. Instead, if we assume a distribution on suboptimality, i.e. given a trajectory of cost $c(\hat{\xi})$, we know that a trajectory of cost $c(\xi') \in [\frac{c(\hat{\xi})}{1+\delta}, c(\hat{\xi})]$ is unsafe with probability p_k, we can then use these values of p_k to solve Problem 5.

5 Analysis

In this section, we provide theoretical analysis on our constraint learning algorithm. In particular, we analyze the limits of what constraint states can be learned guaranteed unsafe for both the gridded and parametric cases (Sections 5.1 and 5.3) as well as the conditions under which our algorithm is guaranteed to learn an inner approximation of the safe and unsafe sets (Sections 5.2 and 5.4). For ease of reading, the proofs and some remarks are omitted and can be found in the appendix.

We begin with an overview of the theoretical results:

- Theorem 1 shows that all states that can be guaranteed unsafe must lie within some distance to the boundary of the unsafe set. Corollary 1 shows that the set of guaranteed unsafe states shrinks to a subset of the boundary of the unsafe set when using a continuous demonstration directly to learn the constraint.
- Corollary 2 shows that under assumptions on the alignment of the grid and unsafe set for the discrete time case, the guaranteed learned unsafe set is a guaranteed inner approximation of the true unsafe set.
- For continuous trajectories that are then discretized, Theorem 3 shows us that the guaranteed unsafe set can be made to contain states on the interior of the unsafe set, but at the cost of potentially labeling states within some distance outside of the unsafe set as unsafe as well.
- Theorem 4 shows that for the parametric case, all states that can be guaranteed unsafe must be implied unsafe by the states within some distance to the boundary of the unsafe set and the parameterization.
- Theorem 5 shows that for the discrete time case, the guaranteed safe and guaranteed unsafe sets are inner approximations of the true safe and unsafe sets, respectively. For the continuous time case, the recovered sets are inner approximations of a padded version of the true sets.

5.1 Learnability

We provide analysis on the learnability of unsafe sets, given the known constraints and cost function. Most analysis assumes unsafe sets defined over the state space: $\mathcal{A} \subseteq \mathcal{X}$, but we extend it to the feature space in Corollary 2. We provide some definitions and state a result bounding ${\mathcal{G}^z_{\neg s}}^*$, the set of all states that **can** be learned guaranteed unsafe. We first define the signed distance:

Definition 1. Signed distance. *Signed distance from point $p \in \mathbb{R}^m$ to set $\mathcal{S} \subseteq \mathbb{R}^m$, $\mathsf{sd}(p, \mathcal{S}) = -\inf_{y \in \partial\mathcal{S}} \|p-y\|$ if $p \in \mathcal{S}$; $\inf_{y\in\partial\mathcal{S}} \|p-y\|$ if $p \in \mathcal{S}^c$.*

Theorem 1. Learnability (discrete time). *For trajectories generated by a discrete time dynamical system satisfying $\|x_{t+1} - x_t\| \leq \Delta x$ for all t, the set of learnable guaranteed unsafe states is a subset of the outermost Δx shell of the unsafe set: ${\mathcal{G}^z_{\neg s}}^* \subseteq \{x \in \mathcal{A} \mid -\Delta x \leq \mathsf{sd}(x, \mathcal{A}) \leq 0\}$ (see Section A.1, Figure A.1 for an illustration).*

Corollary 1. Learnability (continuous time). *For continuous trajectories $\xi(\cdot) : [0,T] \to \mathcal{X}$, the set of learnable guaranteed unsafe states shrinks to the boundary of the unsafe set: ${\mathcal{G}^z_{\neg s}}^* \subseteq \{x \in \mathcal{A} \mid \mathsf{sd}(x, \mathcal{A}) = 0\}$.*

Depending on the cost function, ${\mathcal{G}^z_{\neg s}}^*$ can become arbitrarily small: some cost functions are not very informative for recovering a constraint. For example, the path length cost function used in many of the experiments (which was chosen due to its common use in the motion planning community), prevents any lower-cost sub-trajectories from being sampled from straight sub-trajectories. The overall control authority that we have on the system also impacts learnability: the more controllable the system, the more of the Δx shell is reachable. In particular, a necessary condition for any unsafe states to be learnable from a demonstration of length $T+1$ starting from x_0 and ending at x_T is for there to be more than one trajectory which steers from x_0 to x_T in $T+1$ steps while satisfying the dynamics and control constraints.

5.2 Conservativeness

We discuss conditions on $\mathcal{A}$ and discretization which ensure our method provides a conservative estimate of $\mathcal{A}$. For analysis, we assume $\mathcal{A}$ has a Lipschitz boundary (Dacorogna (2015)). We begin with notation (an explanatory illustration is in Section A.2, Figure A.2):

Definition 2. Normal vectors. *Denote the outward-pointing normal vector at a point $p \in \partial\mathcal{A}$ as $\hat{n}(p)$. Furthermore, at non-differentiable points on $\partial\mathcal{A}$, $\hat{n}(p)$ is replaced by the set of normal vectors for the sub-gradient of the Lipschitz function describing $\partial\mathcal{A}$ at that point (Allaire et al. (2016)).*

Definition 3. γ-offset padding. *Define the γ-offset padding $\partial\mathcal{A}_\gamma$ as: $\partial\mathcal{A}_\gamma = \{x \in \mathcal{X} \mid x = y + d\hat{n}(y), d \in [0,\gamma], y \in \partial\mathcal{A}\}$.*

Definition 4. γ-padded set. *We define the γ-padded set of the unsafe set $\mathcal{A}$, $\mathcal{A}(\gamma)$, as the union of the γ-offset padding and $\mathcal{A}$: $\mathcal{A}(\gamma) \doteq \partial\mathcal{A}_\gamma \cup \mathcal{A}$.*

Definition 5. Maximum grid size. *Let $R(z_i)$ be the radius of the smallest ball which contains grid cell z_i: $R(z_i) = \min_r \min_{x_i} r$, subject to $z_i \subseteq B_r(x_i)$, for some optimal center x_i.*

Furthermore, let R^* be the radius of the smallest ball which contains each grid cell $z_i, i = 1, \ldots, G$: $R^* = \max(R(z_1), \ldots, R(z_G))$.

We also introduce the following assumption, which is illustrated in Figure A.3 for clarity:

Assumption 1: The unsafe set $\mathcal{A}$ is aligned with the grid (i.e. there does not exist a grid cell z containing both safe and unsafe states in its interior).

Theorem 2. Discrete time conservative recovery of unsafe set. *For a discrete-time system, if Assumption 1 holds, $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}$. If Assumption 1 does not hold, then $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}(R^*)$.*

If we use continuous trajectories directly, the guaranteed learnable set $\mathcal{G}^z_{\neg_s}{}^*$ shrinks to a subset of the boundary of the unsafe set, $\partial\mathcal{A}$ (c.f. Corollary 1). However, if we discretize these trajectories, we show that we can learn unsafe states lying in the interior, at the cost of conservativeness holding only for a padded unsafe set. We then show that a similar result holds when discretizing a continuous trajectory in a feature space. For the following results, we make an additional assumption, which is illustrated in Figure A.4 for clarity:

Assumption 2: The time discretization of the unsafe trajectory $\xi : [0, T] \to \mathcal{X}$, $\{t_1, \ldots, t_N\}, t_i \in [0, T]$, for all i, is chosen such that there exists at least one discretization point in the interior of each cell that the continuous trajectory passes through (i.e. if $\exists t \in [0, T]$ such that $\xi(t) \in z$, then $\exists t_i \in \{t_1, \ldots, t_N\}$ such that $\xi(t_i) \in z$.

Theorem 3. Continuous-to-discrete time conservativeness. *The following results hold for continuous time systems:*

1. *Suppose that both Assumptions 1 and 2 hold. Then, the learned guaranteed unsafe set $\mathcal{G}^z_{\neg_s}$, defined in Section 4.4.1, is contained within the true unsafe set $\mathcal{A}$.*
2. *Suppose that only Assumption 2 holds. Then, the learned guaranteed unsafe set $\mathcal{G}^z_{\neg_s}$ is contained within the R^*-padded unsafe set, $\mathcal{A}(R^*)$.*
3. *Suppose that neither Assumption 1 nor Assumption 2 holds. Furthermore, suppose that Problems 2, 4, and 5 are using M sub-trajectories sampled with Algorithm 1 as unsafe trajectories, and that each sub-trajectory is defined over the time interval $[a_i, b_i], i = 1, \ldots, M$. Denote $D_\xi([a, b]) \doteq \sup_{t_1 \in [a,b], t_2 \in [t_1,b]} \|\xi(t_1) - \xi(t_2)\|_2$, for some trajectory ξ. Denote $D^* \doteq \max_{i \in \{1,\ldots,M\}} D^*_{\xi_i}([a_i, b_i])$. Then, the learned guaranteed unsafe set $\mathcal{G}^z_{\neg_s}$ is contained within the $D^* + R^*$-padded unsafe set, $\mathcal{A}(D^* + R^*)$.*

Corollary 2. Continuous-to-discrete feature space conservativeness. *Let the feature mapping $\phi(x)$ from the state space to the constraint space be Lipschitz continuous with Lipschitz constant L. Then, the following results hold:*

1. *Suppose both Assumptions 1 and 2 (used in Theorem 3) hold. Then, our method ensures $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}$.*
2. *Suppose only Assumption 2 holds. Then, our method recovers a guaranteed subset of the LR^*-padded unsafe set, $\mathcal{A}(LR^*)$, in the feature space.*
3. *Suppose neither Assumption 1 nor Assumption 2 holds. Then, our method recovers a guaranteed subset of the $L(D^* + R^*)$-padded unsafe set, $\mathcal{A}(L(D^* + R^*))$, where D^* is as defined in Theorem 3.*

5.3 Parametric learnability

In this section, we develop results for learnability of the unsafe set in the parametric case. For clarity, we prove the results for $\mathcal{C} = \mathcal{X}$. We begin with the following notation:

Definition 6. Implied unsafe set. *For some set $\mathcal{B} \subseteq \Theta$, denote*

$$I(\mathcal{B}) \doteq \bigcap_{\theta \in \mathcal{B}} \{x \mid g(x, \theta) \leq 0\} \tag{13}$$

as the set of states that are implied unsafe by restricting the parameter set to $\mathcal{B}$. In words, $I(\mathcal{B})$ is the set of states for which all $\theta \in \mathcal{B}$ mark as unsafe.

The following result states that in discrete time, the learnable set of unsafe states $\mathcal{G}^*_{\neg_s}$ is contained by the set of states which must be implied unsafe by learning that all states in the outer Δx shell of the unsafe set, $\mathcal{A}_{\Delta x}$, are unsafe. Furthermore, in continuous time, the same holds, except the Δx shell is replaced by the boundary of the unsafe set, $\partial\mathcal{A}$.

Theorem 4. Discrete time learnability for parametric constraints. *For trajectories generated by discrete time systems, $\mathcal{G}_{\neg_s} \subseteq \mathcal{G}^*_{\neg_s} \subseteq I(\mathcal{F}_{\Delta x})$, where*

$$\mathcal{F}_{\Delta x} = \{\theta \mid \forall i \in \{1, \ldots, N_s\}, \forall x \in \xi^*_i, g(x, \theta) > 0, \\ \forall x \in \mathcal{A}_{\Delta x}, g(x, \theta) \leq 0\}$$

Corollary 3. Continuous-time learnability for parametric constraints. *For trajectories generated by continuous time systems, $\mathcal{G}_{\neg_s} \subseteq \mathcal{G}^*_{\neg_s} \subseteq I(\mathcal{F}_{\partial\mathcal{A}})$, where*

$$\mathcal{F}_{\partial\mathcal{A}} = \{\theta \mid \forall x \in \xi^*_i, \forall i \in \{1, \ldots, N_s\}, g(x, \theta) > 0, \\ \forall x \in \partial\mathcal{A}, g(x, \theta) \leq 0\}$$

5.4 Parametric conservativeness

We write conditions for conservative recovery of the unsafe set and safe set when solving Problems 3 and 6 for discrete time and continuous time systems.

Theorem 5. Conservative recovery for discrete time systems with parametric constraints. *For a discrete-time system, if M in Problem 6 is chosen to be greater than $\max(M_1, M_2)$, where $M_1 = \max_{x_i \in \xi_s} \max_\theta \max_j (H(\theta)x_i - h(\theta))_j$ and $M_2 = \max_{x_i \in \xi_{\neg_s}} \max_\theta \max_j (H(\theta)x_i - h(\theta))_j$, $\mathcal{G}_{\neg_s} \subseteq \mathcal{A}$ and $\mathcal{G}_s \subseteq \mathcal{S}$.*

Corollary 4. Conservative recovery for continuous time systems with parametric constraints. *For a continuous-time system, where demonstrations are time-discretized as previously discussed, if M is chosen as in Theorem 5, $\mathcal{G}_s \subseteq \mathcal{S}$ and $\mathcal{G}_{\neg_s} \subseteq \mathcal{A}(D^*)$, where D^* is as defined in Theorem 3.*

	Fig. 8, Row 1	Fig. 8, Row 2	Fig. 8, Row 3	Fig. 9	Fig. 10	Fig. 11	Fig. 13	Fig. 15	Fig. 18
Space discretization	0.1	0.25	0.5	1	1	n/a	n/a	n/a	n/a
Number of trajectories sampled	300000	150000	10000	10000	10000	100000	250000	250000	10000
Number of trajectories used	300000	150000	10000	10000	10000	125	15000	1500	500
ε	n/a	n/a	10^{-3}	10^{-3}	10^{-3}	n/a	n/a	n/a	10^{-3}
$\hat{\varepsilon}$	n/a	n/a	10^{-2}	10^{-2}	10^{-2}	n/a	n/a	n/a	10^{-2}
α_c	10^{10}	10^{4}	1	1	1	10^{10}	10^{10}	10^{10}	1
Minimum $\mathcal{L}$ length	n/a	n/a	10^{-10}	10^{-10}	10^{-10}	n/a	n/a	n/a	10^{-10}
D^*	n/a	n/a	7.85	10.5	0.04	n/a	n/a	n/a	n/a
R^*	0.07	0.35	0.35	0.70	0.005	n/a	n/a	n/a	n/a

Table 2. Parameters used for each experiment.

Experiment	Time (sampling trajectories)	Time (constraint recovery)
Single integrator, U-shape, gridded; Fig. 8, Row 1	11.5 min	3 min
Double integrator, gridded; Fig. 8, Row 2	4.5 min	4.5 min
Dubins' car, gridded; Fig. 8, Row 3	2 hrs	4 min
Dubins' car, suboptimal, gridded; Fig. 9	1 hr	2 min
Dubins' car, feature space, gridded; Fig. 10	30 min	4 min
Single integrator, U-shape, parametric; Fig. 11	1.5 min	27.3 seconds
7-DOF arm; Fig. 11	12.5 min	1.2 seconds
7-DOF arm, suboptimal; Fig. 11	9 min	1.2 seconds
Quadrotor; Fig 18	8.5 min	11.9 seconds

Table 3. Approximate runtimes for each experiment.

6 Evaluations: Gridded formulation

In this section and the next (Section 7), we evaluate the effectiveness of both our gridded and parametric variants of the constraint recovery problem on a variety of examples. Experiment parameters and approximate runtimes for all examples can be found in Tables 2 and 3. All experiments were conducted on a 4-core 2017 Macbook Pro with a 3.1 GHz Core i7 processor. All code was implemented in MATLAB.

We evaluate the gridded variant of our method on a variety of constraint recovery problems in this section. In particular, we provide examples showing the effectiveness of using unsafe trajectories to reduce the ill-posedness of the constraint-recovery problem (Section 6.1), that our method has advantages over inverse reinforcement learning (Section 6.2), that our method can be applied for discrete-time, continuous-time, linear, and nonlinear system dynamics (Section 6.3), that our method can be adapted to work with suboptimal demonstrations (Section 6.4), and that our method can also learn constraints in arbitrary feature spaces (Section 6.5).

6.1 Version space example

Consider a simple 5×5 8-connected grid world in which the tasks are to go from a start to a goal, minimizing Euclidean path length while staying out of the unsafe "U-shape", the outline of which is drawn in black (Fig. 5). Four demonstrations are provided, shown in Fig. 5 on the far left. Initially, the version space contains 2^{25} possible unsafe sets.

	1	2	3	4
Safe	262144	4096	1024	256
Safe & unsafe	11648	48	12	3

Table 4. Number of consistent unsafe sets, varying the number of demonstrations, using/not using unsafe trajectories (c.f. the example in Section 6.1).

Each safe trajectory of length T reduces the version space at most by a factor of 2^T, invalidating at most $2^{25} - 2^{25-T}$ possible unsafe sets. Unsafe trajectories are computed by enumerating the set of trajectories going from the start to the goal at lower cost than the demonstration. The numbers of unsafe sets consistent with the safe and unsafe trajectories for varying numbers of safe trajectories are given in Table 4.

Ultimately, it is impossible to distinguish between the three unsafe sets on the right in Fig. 5. This is because there exists no task where a trajectory with cost lower than the demonstration can be sampled which only goes through one of the two uncertain states. Further, though the uncertain states are in the Δx shell of the constraint, due to the limitations of the cost function, we can only learn a subset of that shell (c.f. Theorem 1).

There are two main takeaways from this experiment. First, by generating unsafe trajectories, we can reduce the uncertainty arising from the ill-posedness of constraint learning: after 4 demonstrations, using unsafe demonstrations enables us to reduce the number of possible constraints by nearly a factor of 100, from 256 to 3. Second, due to limitations in the cost function, it may be impossible to recover a unique unsafe set, but the version space can be reduced substantially by sampling unsafe trajectories.

6.2 Comparison with inverse reinforcement learning

In this section, we illustrate some advantages of explicitly learning a hard constraint from demonstrations over learning a softened penalty through two examples.

6.2.1 Gridded example Consider the grid world in Figure 6(a), where the available actions at each state are to move up, down, left, right (except when doing so goes out of bounds), and an "exit" action, which takes the agent to a terminal state. In this setting, the objective of the demonstrator is to minimize the path length to the goal (green square) while avoiding the unsafe set (red squares), and we are given one demonstration doing so (see Figure 6(a)).

Suppose that as the learner, we know the objective is path length (see Figure 6(b)) and we also know the goal (see Figure 6(c)), and we would like to learn the

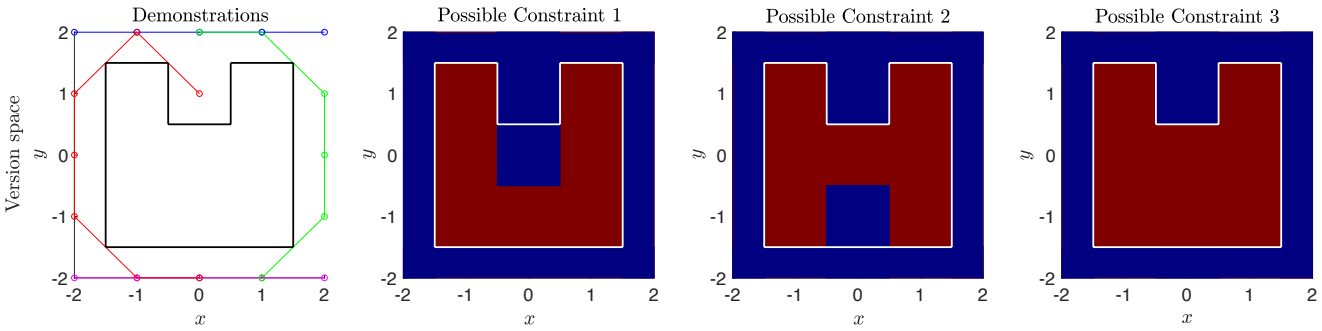

Figure 5. ***Leftmost****: Demonstrations and unsafe set.* ***Rest****: Set of possible constraints. Postulated unsafe cells are plotted in red, safe states in blue.*

unknown red constraint by representing it as an unknown penalty component in the reward function and learning it via inverse reinforcement learning (Ng and Russell (2000)). We can write this problem in the framework of IRL by representing the grid world as a deterministic, finite horizon Markov decision process $\langle S, A, P, R, T\rangle$, where the state space S, action space A, and transition probabilities P are as described in the previous paragraph, and T is the time horizon ($T = 11$ for this problem). The reward function $R(s, a)$ is assumed to have a known component $R_{\text{goal}}(s, a) + R_{\text{path}}(s, a)$ and unknown component $R_{\text{unsafe}}(s, a)$, which is to be learned from the demonstrations. Specifically, the path length aspect of the cost function is modeled by a small negative reward upon reaching each state (see Figure 6(b)), $R_{\text{goal}}(s, a)$ is zero except for a large positive reward obtained by taking the exit action at the goal state (see Figure 6(c)), and we take the reward function to be of the form $R(s, a) = R_{\text{goal}}(s, a) + R_{\text{path}}(s, a) + R_{\text{unsafe}}(s, a)$. One penalty function $R_{\text{unsafe}}(s, a)$ which is consistent with the demonstration and the known reward function component is shown in Figure 6(d) (here, the numerical labels on each state correspond to the reward obtained when taking an action that reaches that state). This is because by using this penalty, there exists no trajectory that achieves a larger cumulative reward than the demonstration, under the combined reward function (Figure 6(e)).

However, using this learned penalty when starting from the bottom right state leads to an optimal path which is unsafe (Figure 6(f)), as the learned penalty is only consistent with the observed demonstrations but does not necessarily adequately enforce the constraint starting from novel initial states. On the other hand, using our method, by sampling lower-cost trajectories, we can learn that each state on the middle row except for the leftmost state is guaranteed unsafe. Using this learned constraint and planning a path starting from the bottom right state leads to a path which avoids the unsafe set. Similarly, the learned penalty will not necessarily be valid when changing the known component of the reward function (i.e. the goal state) because the learned penalty values will depend on the values of the known component, while the learned constraint is agnostic to the known component and will transfer across different known reward functions, and unsafe paths can be planned using the learned penalty (Figure 6(f)-(g)).

Overall, the key takeaways of this example are to show that representing a constraint as a reward penalty may lead to unsafe behavior when planning trajectories from new start states or to new goal states, while explicitly learning the constraint transfers more reliably across tasks.

6.2.2 Parametric example Consider the problem illustrated in Figure 7, where the demonstrator's objective is to minimize path length while avoiding the red obstacle and satisfying input constraints: that is, the demonstrator solves Problem 1, where the obstacle avoidance constraint is encoded in the unsafe set $\mathcal{A} = \{x \mid [I_{2\times 2}, -I_{2\times 2}]^\top x \leq \theta\}$, where $\theta = [2, 2, 2, 2]^\top$ and the path length objective is encoded in $c_\Pi(\xi_x, \xi_u) = \sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_2^2$. We consider two variants of the inverse optimization problem: in the first variation, we solve Problem 6 to directly recover the parameters θ defining the unknown constraint. In the second variation, we modify Problem 1 to soften the obstacle avoidance constraint to a cost penalty, posing the problem as:

$$\begin{aligned} \underset{\xi_x, \xi_u}{\text{minimize}} \quad & \sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_2^2 + \lambda \sum_{t=1}^{T} \mathbf{1}_{x_t \notin \mathcal{S}} \\ \text{s.t.} \quad & \phi(\xi_x, \xi_u) \in \bar{\mathcal{S}} \subseteq \bar{\mathcal{C}} \\ & \phi_\Pi(\xi_x, \xi_u) \in \mathcal{S}_\Pi \subseteq \mathcal{C}_\Pi \end{aligned} \tag{14}$$

where the constraints encode the input and start/goal constraints, $\mathbf{1}_{(\cdot)}$ denotes the indicator function for the event $(\cdot)$, and λ is a nonnegative penalty coefficient. In this variation of the learning problem, we assume that $\mathcal{S}$ is known, and we only aim to learn a suitable penalty coefficient λ which makes the demonstrations globally-optimal. At this point, we should also note that it can be challenging to determine a suitable cost penalty parameterization; for example, while $\lambda \sum_{t=1}^{T} \|x_t\|_\infty$ may appear to be a good penalty parameterization, we could not find a value of λ for this parameterization that replicated the demonstrated behavior presented in Figure 7.

By solving Problem 6 using the two provided demonstrations and sampled lower-cost trajectories, θ can be learned exactly, and the guaranteed safe/unsafe sets match with the true safe/unsafe sets ($\mathcal{G}_{\neg s} = \mathcal{A}$ and $\mathcal{G}_s = \mathcal{S}$). On the other hand, choosing $\lambda = 0.15$ in (14) is a sufficiently large penalty to make the solution of (14) match with the demonstrations. However, like the example in Figure 6, planning new trajectories from different start or goal states can lead to unsafe trajectories under the learned cost function (see Figure 7, right). Furthermore, the notion of guaranteed unsafeness of a state is not meaningful for the softened case, as states that are avoided ("unsafe") for one pair of start/goal

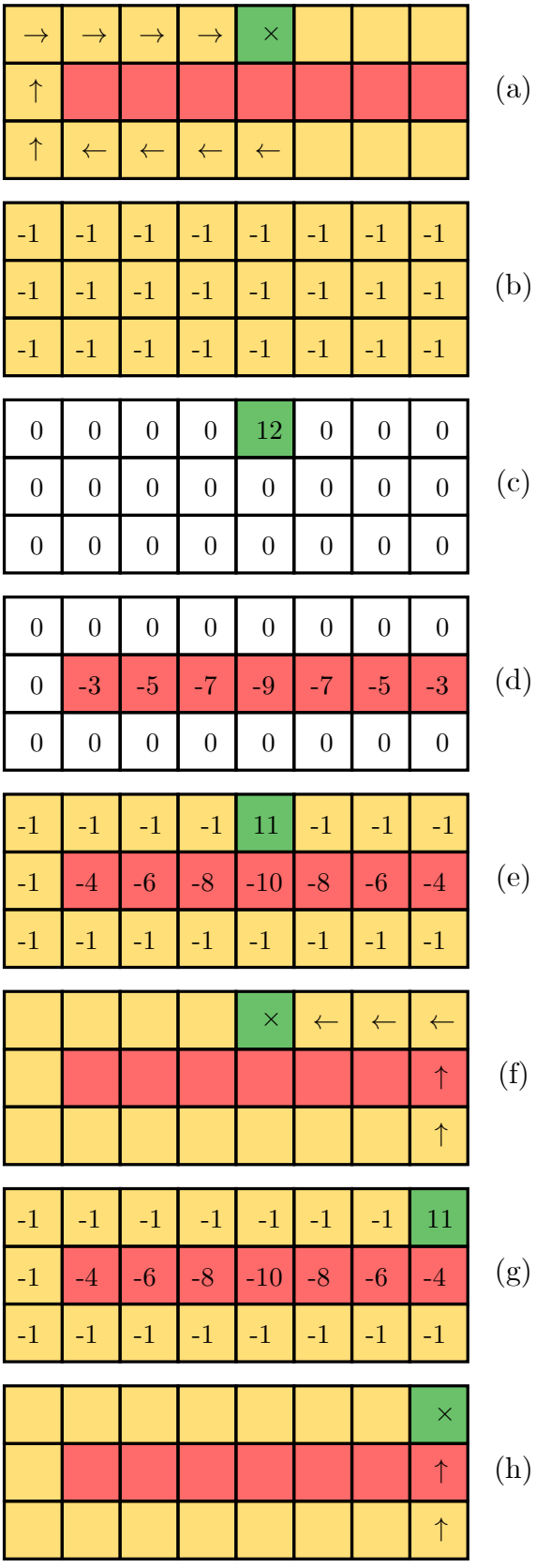

Figure 6. *IRL comparison (gridded). (a) Demonstration. (b) Path length component of reward function $r_{path}(x)$ (numbers indicate the reward obtained upon reaching a state). (c) Goal component of reward function $r_{goal}(x)$. (d) A consistent softened constraint reward function. (e) Combined reward function. (f) Unsafe optimal trajectory from a new initial condition under the combined reward function. (g) Combined reward function for a different goal. (h) Unsafe optimal trajectory when planning with a different goal.*

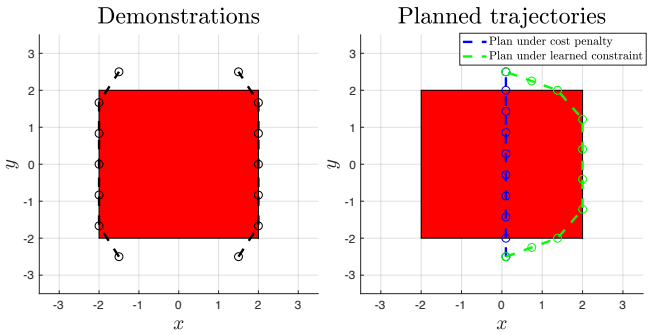

Figure 7. *IRL comparison (parametric). **Left**: We are given two demonstrations avoiding a red obstacle. **Right**: Paths planned with the cost penalty may be unsafe, whereas trajectories planned with the learned constraint remain safe.*

states may be visited ("safe") for a different pair, provided it is less costly to receive a penalty for violating the constraint compared to planning a higher-cost trajectory that satisfies the constraint.

6.3 Dynamics and discretization

Experiments in Fig. 8 show that our method can be applied to several types of system dynamics, can learn non-convex/multiple unsafe sets, and can use continuous trajectories. All unsafe sets $\mathcal{A}$ are open sets. We solve Problems 4 and 5, with an energy function promoting smoothness by penalizing squared deviations of the occupancy of a grid cell z_i from its 4-connected neighbors $N(z_i)$: $\sum_{i=1}^{G}\sum_{z_j\in N(z_i)}\|\mathcal{O}(z_i)-\mathcal{O}(z_j)\|_2^2$. In all experiments, the mean squared error (MSE) is computed as $\frac{1}{G}\sqrt{\sum_{i=1}^{G}\|\mathcal{O}(z_i)^*-\mathcal{O}(z_i)\|_2^2}$, where $\mathcal{O}(z_i)^*$ is the ground truth occupancy. The demonstrations are color-matched with their corresponding number on the x-axis of the MSE plots. For experiments with more demonstrations, only those causing a notable change in the MSE were color-coded. The learned guaranteed unsafe states $\mathcal{G}^z_{\neg s}$ are colored red on the left column.

First, we recover a non-convex "U-shaped" unsafe set in the state space using trivial 2D single-integrator dynamics: $x=[\chi,y]^\top$, $x_{t+1}=x_t+u_t$, with control constraints $\|u_t\|\le 0.5$, for all t. The demonstrator minimizes $c(\xi_x,\xi_u)=\sum_{t=1}^{T-1}\|u_t\|_2^2$. The results are shown in row 1 of Fig. 8. The solutions to both Problems 5 and 4 return reasonable results, and the solution of Problem 4 achieves zero error.

The second row shows learning two polyhedral unsafe sets in the state space with 4D double integrator linear dynamics: $x=[\chi,\dot\chi,y,\dot y]^\top$, where $x_{t+1}=Ax_t+Bu_t$, where $A=\exp\left(\text{diag}\left(\begin{bmatrix}0&1\\0&0\end{bmatrix},\begin{bmatrix}0&1\\0&0\end{bmatrix}\right)\right)$, $B=\int_0^1\exp(A\tau)d\tau\begin{bmatrix}0&1&0&1\end{bmatrix}^\top$, with control constraints $|u_t|\le[20,10]^\top$, for all t. The demonstrator minimizes $c(\xi_x,\xi_u)=\sum_{t=1}^{T-1}\|x_{t+1}-x_t\|_2^2$. The learning procedure yields similar results. We note the linear interpolation of some demonstrations in row 1 and 2 enter $\mathcal{A}$; this is because both sets of dynamics are in discrete time and only the discrete waypoints must stay out of $\mathcal{A}$.

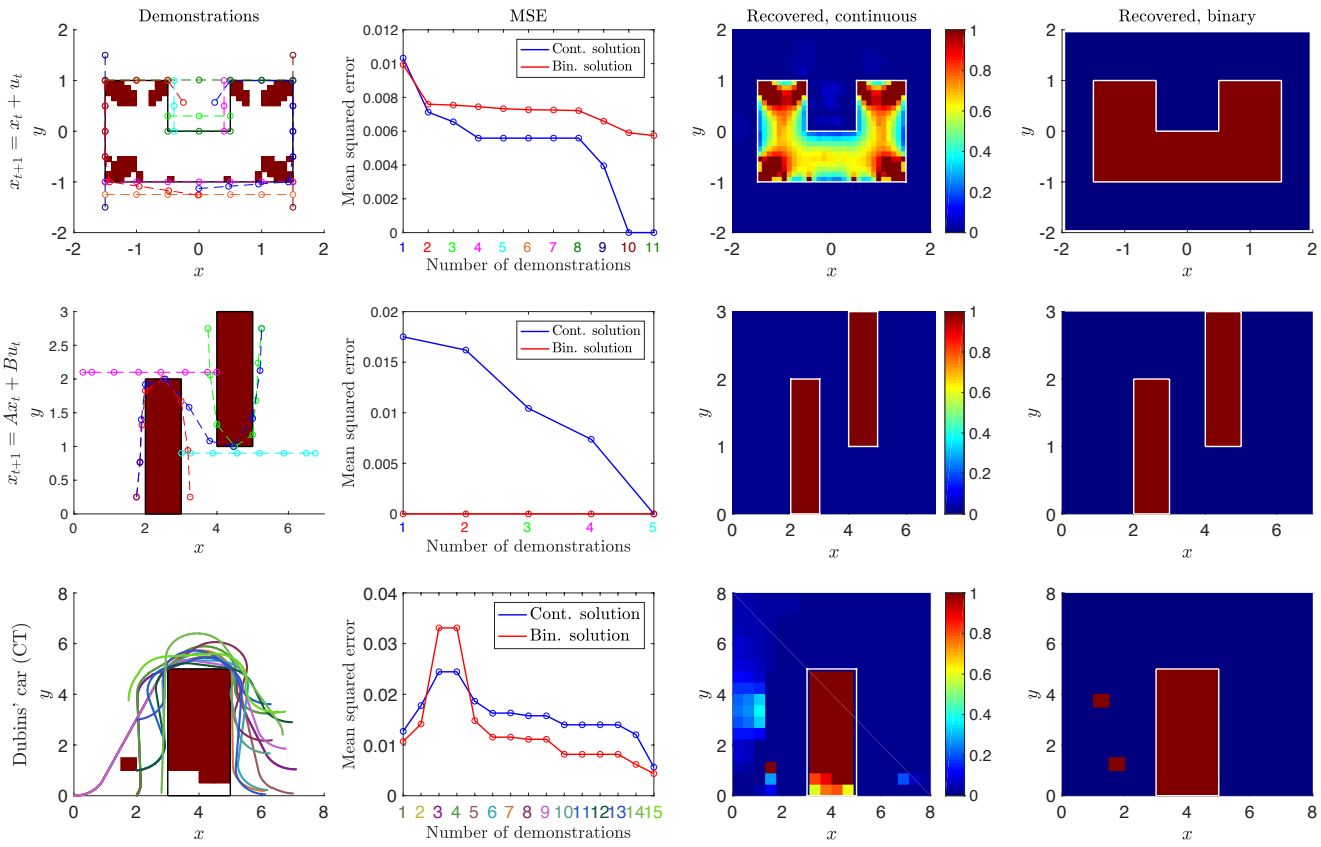

Figure 8. *Results for various dynamical systems and time discretization.* ***Rows (top-to-bottom)****: Single integrator; double integrator; Dubins' car (CT).* ***Columns (left-to-right)****: Demonstrations are plotted together with the outline of the true unsafe set $\mathcal{A}$, and the learned guaranteed unsafe set $\mathcal{G}^z_{\neg s}$ is overlaid (the red cells); mean squared error between the output of Problem 4 or Problem 5 and the ground truth; Problem 5 solution, using all demonstrations; Problem 4 solution, using all demonstrations.*

The third row shows learning a polyhedral unsafe set in the state space, with time-discretized continuous, nonlinear Dubins' car dynamics, which has a 3D state $x \doteq \begin{bmatrix}\chi & y & \theta\end{bmatrix}^\top$ and dynamics $\dot{x} = [\cos(\theta), \sin(\theta), u]^\top$ with control constraints $|u| \leq 1$. The demonstrator minimizes $c(\xi_x, \xi_u) = \sum_u \tau_{u_i}$, where τ_{u_i} is the total time duration of applied control input (i.e. the time it took to go from start to goal). These dynamics are more constrained than the previous cases, so sampling lower cost trajectories becomes more difficult, but despite this we can still achieve near zero error solving Problem 4. Some over-approximation results from some sampled unsafe trajectories entering regions not covered by the safe trajectories, i.e. there are red guaranteed learned unsafe cells outside the true unsafe set. For example, in the right column, the two red blocks to the top left of $\mathcal{A}$ are generated by lower-cost trajectories that trade off the increased cost of entering these grid cells by entering $\mathcal{A}$. This phenomenon is consistent with Theorem 3; we recover a set that is contained within a $D^* + R^*$-padding of $\mathcal{A}$ (here, $D^* + R^* = 8.2$). Learning curve spikes occur when over-approximation occurs.

Overall, we note that for the gridded case, $\mathcal{G}^z_{\neg s}$ tends to be a significant underapproximation of $\mathcal{A}$ due to the chosen cost function and limited demonstrations. For example, in row 1 of Fig. 8, $\mathcal{G}^z_{\neg s}$ cannot contain the portion of $\mathcal{A}$ near long straight edges, since there exists no shorter path going from any start to any goal with only one state within that region. For row 3 of Fig. 8, we learn less of the bottom part of $\mathcal{A}$ due to most demonstrations' start and goal locations making it harder to sample feasible control trajectories going through that region; with more demonstrations, this issue becomes less pronounced. In Section 7.1, we discuss how using a constraint parameterization can reduce the gap between $\mathcal{G}^z_{\neg s}$ and $\mathcal{A}$.

6.4 Suboptimal human demonstrations

We demonstrate our method on suboptimal demonstrations collected via a driving simulator, using a car model with CT Dubins' car dynamics identical to those described in Section 6.3. Human steering commands were recorded as demonstrations, where the task was to navigate around the orange box and drive between the trees (Fig. 9). For a demonstration of cost c, trajectories with cost less than $0.9c$ were believed unsafe with probability 1. Trajectories with cost c' in the interval $[0.9c, c]$ were believed unsafe with probability $1 - ((c' - 0.9c)/0.1c)$. MSE for Problem 5 is shown in Fig. 9 (Problem 4 is not solved since the probabilistic interpretation is needed). For this problem, D^* is 10 seconds and the unsafe set is grid-aligned; hence, despite suboptimality, the learned guaranteed unsafe set is a subset of $\mathcal{A}(D^*)$. While the MSE is highest here of all experiments, this is expected, as trajectories may be incorrectly labeled safe/unsafe with some probability.

6.5 Feature space constraint

We demonstrate that our framework is not limited to the state space by learning a constraint in a feature space. Consider

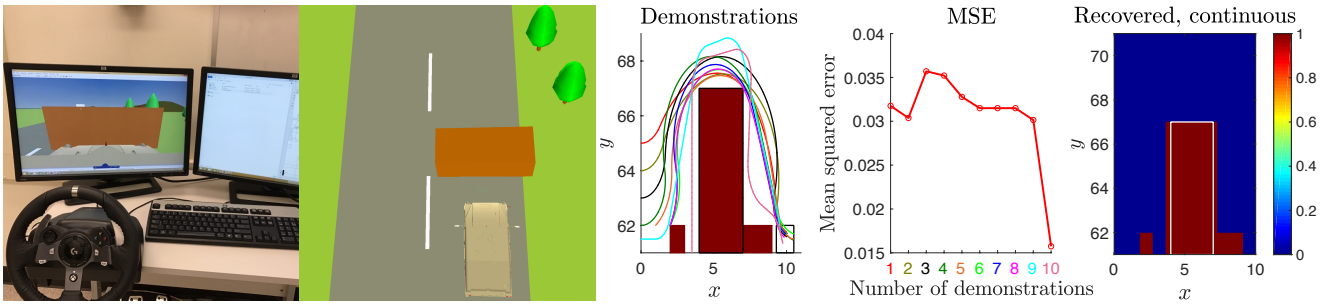

Figure 9. *Suboptimal demonstrations:* ***left****: setup,* ***center****: demonstrations,* $\mathcal{A}$, $\mathcal{G}^z_{\neg s}$, ***center-right****: MSE,* ***right****: solution to Problem 5.*

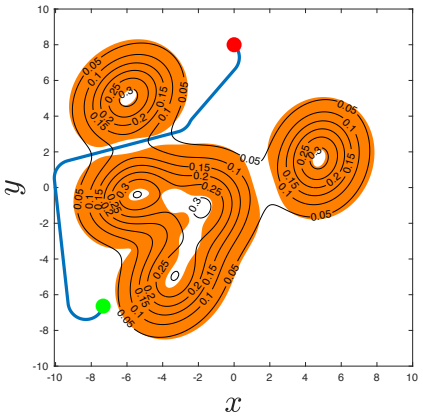

Figure 10. *Feature space constraint recovery. Unsafe set in the constraint space* $\mathcal{A}$ *is plotted in orange. The single demonstration is overlaid (red: start, green: goal). Terrain isocontours* $L(x) = const$ *are overlaid.*

the scenario of planning a safe path for a mobile robot with identical continuous Dubins' car dynamics through hilly terrain, where the magnitude of the terrain's slope is given as a feature map (i.e. $\phi(x) = \|\partial L(\hat{x})/\partial \hat{x}\|_2$, where $\hat{x} = [\chi \; y]^\top$ and $L(\hat{x})$ is the elevation map). The robot will slip if the magnitude of the terrain slope is too large, so we generate a demonstration which obeys the ground truth constraint $\phi(x) < 0.05$; hence, the ground truth unsafe set is $\mathcal{A} \doteq \{x \mid \phi(x) \geq 0.05\}$. From one safe trajectory (Fig. 10) generated by RRT* (Karaman and Frazzoli (2010)) and gridding the feature space as $\{0, 0.005, \ldots, 0.145, 0.15\}$, we recover the constraint $\phi(x) < 0.05$ exactly.

This example shows how using using a feature parameterization can benefit the sample complexity of our method; in the next section, we show that by using a parameterization, the constraint space gridding used so far can be eliminated to improve our method's scalability.

7 Evaluations: Parametric

We evaluate the parametric variant of our method on a variety of constraint recovery problems in this section. In particular, we provide examples showing the effect of using a constraint parameterization on sample complexity and guaranteed learnability (Section 7.1), that our method can be applied to learn a high-dimensional pose constraint (Sections 7.2.1 and 7.2.2), and that our method can work with trajectories generated by high-dimensional dynamics (Section 7.3).

7.1 Comparison to gridded formulation

To demonstrate the advantages provided by assuming a parameterization, we replicate the experiment in the first row of Figure 8, assuming that $\mathcal{A}$ can be represented as the union of three axis-aligned boxes. The results are displayed in Figure 11. Here, the same grid points are queried for guaranteed safeness or unsafeness as described in Section 4.5.

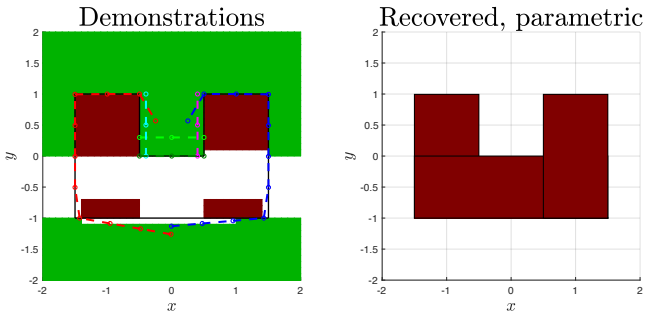

Figure 11. *Replicating row 1 of Figure 8 using a three-box parameterization.* ***Left****:* $\mathcal{G}_{\neg s}$ *is shaded in red and* $\mathcal{G}_s$ *is shaded in green. Demonstrations are overlaid and color-coded to match with row 1 of Figure 8.* ***Right****: Recovered constraint using a variant of Problem 6.*

Compared to the gridded approach, the true unsafe set can be exactly recovered with just six of the original eleven demonstrations (demonstrations 1-5 and 8 in row 1 of Figure 8). This improved sample complexity arises from the fact that our method can extrapolate that some unseen states are safe or unsafe using the parameterization. On the contrary, in the gridded formulation, each grid cell is independent and learning that some cell is unsafe can never imply that another cell is unsafe; only learning that a cell is safe can imply that another cell is unsafe.

Note that compared to Figure 8, a non-trivial $\mathcal{G}_s$ containing states not explicitly covered by demonstrations can now be recovered. Furthermore, $\mathcal{G}_{\neg s}$ covers a larger fraction of the true unsafe set than compared to the gridded approach. As just discussed, this arises from the fact that given the parameterization and some guaranteed unsafe states, other states can be implied unsafe. As a result, $\mathcal{G}_{\neg s}$ expands to include the set of states which must be unsafe to be compatible with the safe and unsafe trajectories and the parameterization as well.

7.2 6D pose constraint for a 7-DOF robot arm

7.2.1 Optimal demonstrations In this example, we learn a six-dimensional hyper-rectangular pose constraint for the

end effector of a 7-DOF Kuka iiwa arm. One example of such a setting is when the robot is asked to pick up a cup and bring it to a human, all while ensuring the cup's contents do not spill (angle constraint) and proxemics constraints (i.e. the end effector never gets too close to the human) are satisfied (position constraint). To this end, the end effector orientation (parametrized in Euler angles) is constrained to satisfy $(\alpha, \beta, \gamma) \in [\underline{\alpha}, \bar{\alpha}] \times [\underline{\beta}, \bar{\beta}] \times [\underline{\gamma}, \bar{\gamma}] = [-\pi, \pi] \times [-\frac{\pi}{60}, \frac{\pi}{60}] \times [-\frac{\pi}{60}, \frac{\pi}{60}]$, while the end effector position is constrained to lie in $(x, y, z) \in [\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}] \times [\underline{z}, \bar{z}] = [-0.51, 0.51] \times [-0.3, 1.1] \times [-0.51, 0.51]$ (these sets are displayed in Figure 12). Seven demonstrations (see Figure 12) optimizing $c(\xi_x, \xi_u) = \sum_{i=1}^{T-1} \|x_{i+1} - x_i\|_2^2$, where $x = [\theta^1, \theta^2, \theta^3, \theta^4, \theta^5, \theta^6, \theta^7]^\top$, are generated by solving the nonlinear trajectory optimization problem using IPOPT (Wächter and Biegler (2006)) for the kinematic, discrete-time model in 7D joint space:

$$\theta_{t+1}^i = \theta_t^i + u_t^i, \quad i = 1, \ldots, 7 \tag{15}$$

Furthermore, for each demonstration $T = 6$ and control constraints $u_t \in [-2, 2]^7$, for all t. Note that the demonstrations push up against the position constraint, since the trajectory minimizing joint-space path length without the position constraint is an arc that exceeds the bounds of the position constraint; the position constraint ends up increasing the cost by truncating that arc.

The constraint is recovered with Problem 6, where $H(\theta) = [I, -I]^\top$ and $h(\theta) = \theta = [\bar{x}, \bar{y}, \bar{z}, \bar{\alpha}, \bar{\beta}, \bar{\gamma}, \underline{x}, \underline{y}, \underline{z}, \underline{\alpha}, \underline{\beta}, \underline{\gamma}]^\top$. The initial search space for the parameters θ is constrained to $[-1.5, 1.5] \times [-1.5, 1.5] \times [-1.5, 1.5] \times [-\pi, \pi] \times [-\pi, \pi] \times [-\pi, \pi]$.

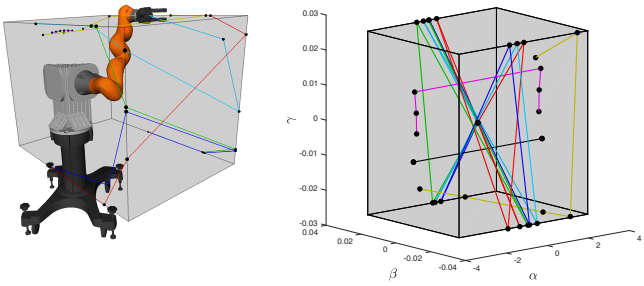

Figure 12. *7-DOF arm setup for optimal demonstrations.* ***Left****: End effector position constraint (gray box).* ***Right****: Euler angle constraint (gray box). The position and angular component of the demonstrations are overlaid and color-coded to match with each other and with the volume statistics in Figure 13.*

Figure 13 shows that as the number of demonstrations increases, $\mathcal{G}_s$ approaches the true safe set $\mathcal{S}$ and $\mathcal{G}_{\neg s}$ approaches the true unsafe set $\mathcal{A}$, respectively. Specifically, in the bottom left plot of Figure 13, as $\mathcal{G}_s \subseteq \mathcal{S}$, $\mathsf{Vol}(\mathcal{G}_s) \to \mathsf{Vol}(\mathcal{S})$ implies convergence to the true safe set. Similarly, in the top left plot of Figure 13, as $\hat{\mathcal{G}}_{\neg s}^c \supseteq \mathcal{A}^c = \mathcal{S}$, $\text{IoU} \doteq \frac{\mathsf{Vol}(\mathcal{A}^c)}{\mathsf{Vol}(\hat{\mathcal{G}}_{\neg s}^c)} \to 1$ implies convergence to the true unsafe set. For the unsafe sets, the volumes are computed for the complement of the unsafe sets for 1) ease of visualization and 2) compared to the unsafe sets, due to the relative sizes of the sets, the change in volume error of the complement is more pronounced as the number of demonstrations increases. The center columns of Figure 13 display a comparison between the safe set and the complement of an inner approximation of the unsafe set, showing that the two match nearly exactly, and the gap in the β direction can be likely reduced with more demonstrations. The right column of Figure 13 displays projections of the recovered safe sets, which match exactly with the true safe sets displayed in Figure 12.

7.2.2 Suboptimal demonstrations Suboptimal demonstrations are recorded using a virtual reality environment, where the demonstrator moves a model of the 7-DOF Kuka iiwa arm from desired start to goal end effector configurations using an HTC Vive (see Figure 16). A visualization of the virtual environment and the test equipment is found in Figure 16. The continuous trajectories are discretized down to $T = 10$ time-steps for lower-cost trajectory sampling. Like for the optimal case, the demonstrator is asked to optimize $c(\xi_x, \xi_u) = \sum_{i=1}^{T-1} \|x_{i+1} - x_i\|_2^2$, for a total of five demonstrations (see Figure 14).

Again, the constraint is recovered with Problem 6, where $H(\theta), h(\theta)$, and the initial search space for θ are as defined for the optimal case. As described in Section 4.5, slack variables are added to ensure feasibility of the problem. Similarly to the example in Section 6.4, for a suboptimal demonstration of cost $\hat{c}$, we only use trajectories of cost less than $0.9\hat{c}$ as unsafe trajectories.

Figure 15 shows that as the number of demonstrations increases, $\mathcal{G}_s$ approaches the true safe set $\mathcal{S}$ and $\mathcal{G}_{\neg s}$ approaches the true unsafe set $\mathcal{A}$, respectively. However, for the suboptimal case, overapproximation of the unsafe set can occur both due to the time discretization from continuous time to discrete time as well not knowing an exact bound on the demonstration suboptimality. The intersection over union (IoU) metric is used to measure how well the complement of the true unsafe set, $\mathcal{G}_s$, matches with the complement of the inner approximation of the learned guaranteed unsafe set, $\mathcal{G}_{\neg s}^c$, and the top left plot in Figure 15 directly plots how the IoU improves with the number of demonstrations. Precisely, we measure $\text{IoU} \doteq \frac{\mathsf{Vol}(\mathcal{S} \cap \mathcal{G}_{\neg s}^c)}{\mathsf{Vol}(\mathcal{S}) + \mathsf{Vol}(\mathcal{G}_{\neg s}^c) - \mathsf{Vol}(\mathcal{S} \cap \mathcal{G}_{\neg s}^c)}$. The low IoU values for lower numbers of demonstrations is due to overapproximation of the unsafe set in the α component (arising from imperfect knowledge of the suboptimality bound); the fifth demonstration, where α takes values near $-\pi, \pi$ greatly reduces this overapproximation and hence improves the IoU value greatly. The middle columns of Figure 15 compares the safe set (the complement of the true unsafe set) and the complement of an inner approximation of the learned unsafe set, showing that the two match well, albeit with a larger gap than for the optimal case. As with the optimal case, this gap can be further reduced with more demonstrations. Note also that the safe set is guaranteed to be an inner approximation of the true safe set, as is the case for all axis-aligned box parameterizations. The right columns of Figure 15 display projections of the recovered safe sets, which closely match the true safe sets.

7.3 3D angular velocity constraint for a 12D quadrotor model

In this example, we learn a three-dimensional hyper-rectangular angular velocity constraint for a quadrotor with twelve-dimensional dynamics (c.f. pages 17-18 of Sabatino (2015)):

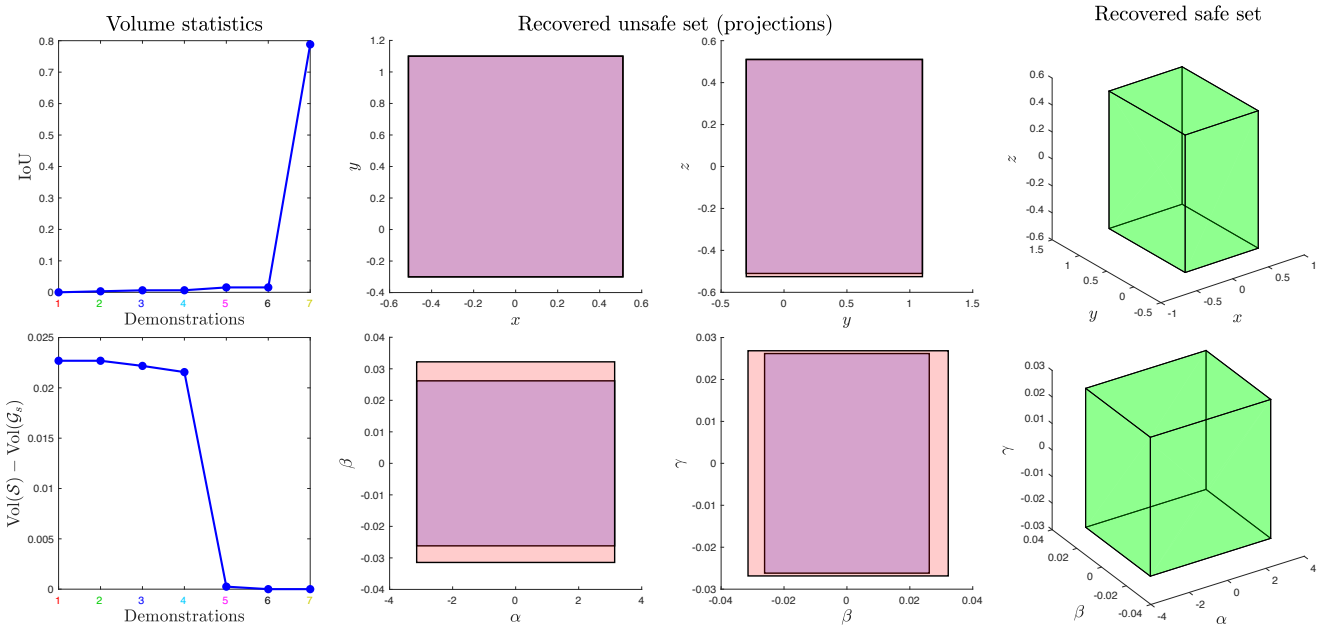

Figure 13. *Constraint recovery for 7-DOF arm, optimal demonstrations.* ***Left, top****: Unsafe set error: error in volume between the* $\hat{\mathcal{G}}^c_{\neg s}$ *and* $\mathcal{A}^c$ *as a function of demonstrations.* ***Left, bottom****: Safe set error: error in volume between* $\mathcal{G}_s$ *and* $\mathcal{S}$*.* ***Center****: projections of* $\mathcal{G}_{\neg s}$ *using all demonstrations.* ***Right****: projections of* $\mathcal{G}_s$ *using all demonstrations.*

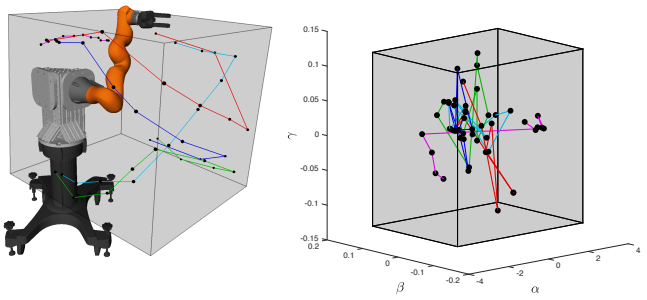

Figure 14. *7-DOF arm setup for suboptimal demonstrations.* ***Left****: End effector position constraint (gray box).* ***Right****: Euler angle constraint (gray box). The position and angular component of the demonstrations are overlaid and color-coded to match with each other and with the volume statistics in Figure 13.*

$$\begin{bmatrix}\dot{\chi}\\ \dot{y}\\ \dot{z}\\ \dot{\alpha}\\ \dot{\beta}\\ \dot{\gamma}\\ \ddot{\chi}\\ \ddot{y}\\ \ddot{z}\\ \ddot{\alpha}\\ \ddot{\beta}\\ \ddot{\gamma}\end{bmatrix} = \begin{bmatrix}\dot{\chi}\\ \dot{y}\\ \dot{z}\\ \dot{\beta}\frac{\sin(\gamma)}{\cos(\beta)} + \dot{\gamma}\frac{\cos(\gamma)}{\cos(\beta)}\\ \beta\cos(\gamma) - \dot{\gamma}\sin(\gamma)\\ \dot{\alpha} + \dot{\beta}\sin(\gamma)\tan(\beta) + \dot{\gamma}\cos(\gamma)\tan(\beta)\\ -\frac{1}{m}[\sin(\gamma)\sin(\alpha) + \cos(\gamma)\cos(\alpha)\sin(\beta)]u_1\\ -\frac{1}{m}[\cos(\alpha)\sin(\gamma) - \cos(\gamma)\sin(\alpha)\sin(\beta)]u_1\\ g - \frac{1}{m}[\cos(\gamma)\cos(\beta)]u_1\\ \frac{I_y - I_z}{I_x}\dot{\beta}\dot{\gamma} + \frac{1}{I_x}u_2\\ \frac{I_z - I_x}{I_y}\dot{\alpha}\dot{\gamma} + \frac{1}{I_y}u_3\\ \frac{I_x - I_y}{I_z}\dot{\alpha}\dot{\beta} + \frac{1}{I_z}u_4\end{bmatrix} \tag{16}$$

For our purposes, we convert the dynamics to discrete time by performing forward Euler integration with discretization time $\delta t = 0.4$ seconds. The state is $x = [\chi, y, z, \alpha, \beta, \gamma, \dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}, \dot{\gamma}]^\top$, and the constants are $g = -9.81\text{m/s}^2$, $m = 1\text{kg}$, $I_x = 0.5\text{kg}\cdot\text{m}^2$, $I_y = 0.1\text{kg}\cdot\text{m}^2$, and $I_z = 0.3\text{kg}\cdot\text{m}^2$.

In this scenario, the quadrotor must avoid a known unsafe set in (χ, y, z): $(\chi, y, z) \notin [-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$ while also ensuring that angular velocities are below a threshold: $(\dot{\alpha}, \dot{\beta}, \dot{\gamma}) \in [\underline{\dot{\alpha}}, \bar{\dot{\alpha}}] \times [\underline{\dot{\beta}}, \bar{\dot{\beta}}] \times [\underline{\dot{\gamma}}, \bar{\dot{\gamma}}]$ (see Figure 17). Specifically, the (x, y, z) unsafe set is known a priori and the $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$ safe set is to be inferred from demonstrations. Two demonstrations optimizing $c(\xi_x, \xi_u) = \sum_{i=1}^{T-1} \|[\chi_{i+1}, y_{i+1}, z_{i+1}, \dot{\alpha}_{i+1}, \dot{\beta}_{i+1}, \dot{\gamma}_{i+1}]^\top - [\chi_i, y_i, z_i, \dot{\alpha}_i, \dot{\beta}_i, \dot{\gamma}_i]^\top\|_2$ (penalizing acceleration and path length) are computed by solving trajectory optimization problems using IPOPT (Wächter and Biegler (2006)), where $T = 25$ and control constraints $[0, -0.02, -0.02, -0.02]^\top \leq u_t \leq [mg, 0.02, 0.02, 0.02]^\top$ for all t.

The constraint is recovered with Problem 6, where $H(\theta) = [I, -I]^\top$ and $h(\theta) = \theta = [\bar{\dot{\alpha}}, \bar{\dot{\beta}}, \bar{\dot{\gamma}}, \underline{\dot{\alpha}}, \underline{\dot{\beta}}, \underline{\dot{\gamma}}]^\top$. The initial search space for the parameters θ is constrained to $[-\frac{\pi}{2}, \frac{\pi}{2}] \times [-\frac{\pi}{2}, \frac{\pi}{2}] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$.

The left column of Figure 18 shows that as the number of demonstrations increases, $\mathcal{G}_s$ approaches the true safe set $\mathcal{S}$ and $\mathcal{G}_{\neg s}$ approaches the true unsafe set $\mathcal{A}$, respectively. As with the optimal 7-DOF arm example, IoU $\doteq \frac{\text{Vol}(\mathcal{A}^c)}{\text{Vol}(\hat{\mathcal{G}}^c_{\neg s})} \to 1$. The center columns of Figure 18 directly plots a comparison between the safe set and the complement of an inner approximation of the unsafe set, showing that the two match exactly. The right column of Figure 18 displays the recovered safe set, which matches exactly with the true safe set displayed in Figure 17.

8 Discussion

In this section, we summarize the main takeaways from the theoretical analysis and experiments:

Learnability of unsafe states: When gridding the constraint space, only grid cells that lie within some distance to the boundary of the unsafe set can be learned guaranteed unsafe.

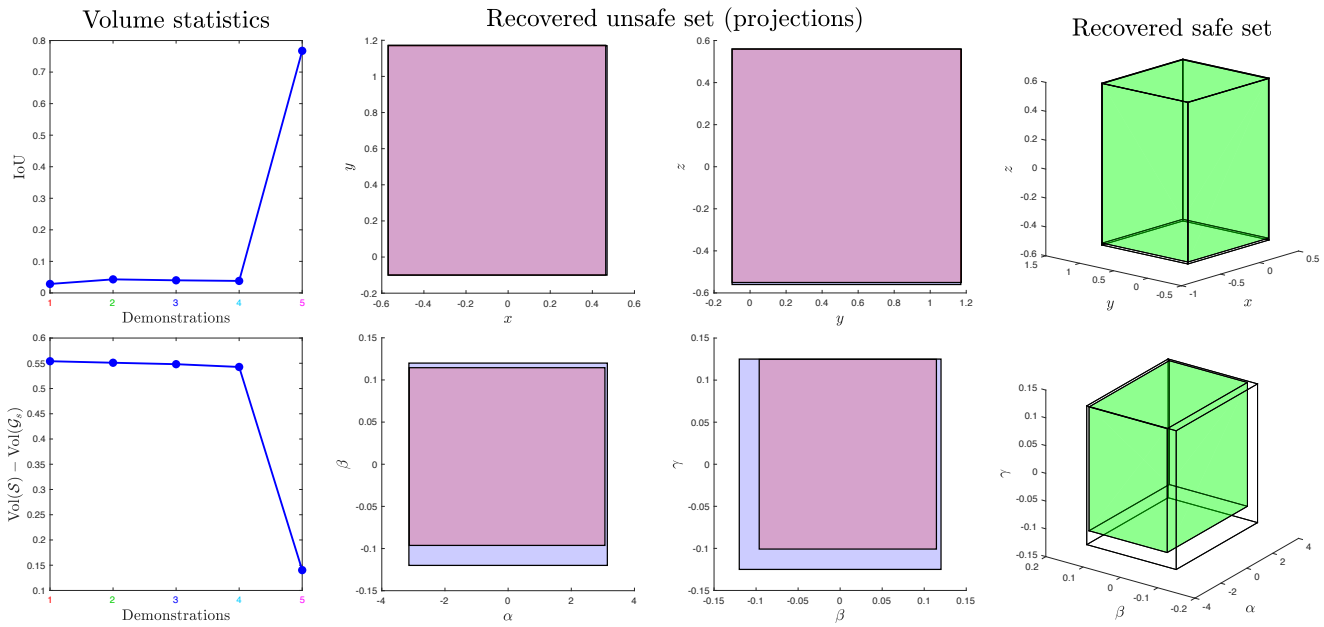

Figure 15. *Constraint recovery for 7-DOF arm, suboptimal demonstrations.* ***Left, top****: Unsafe set error: error in volume between the* $\hat{\mathcal{G}}^c_{\neg s}$ *and* $\mathcal{A}^c$ *as a function of demonstrations.* ***Left, bottom****: Safe set error: error in volume between* $\mathcal{G}_s$ *and* $\mathcal{S}$*.* ***Center****: projections of* $\mathcal{G}_{\neg s}$ *using all demonstrations.* ***Right****: projections of* $\mathcal{G}_s$ *using all demonstrations.*

Figure 16. *VR setup.* ***Top****: VR environment. The green box represents the position constraints. The end effector is commanded to move by selecting and dragging to a desired position with the HTC Vive controllers.* ***Bottom****: Vive hardware.*

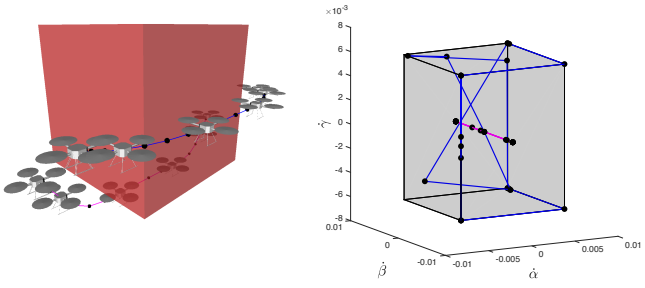

Figure 17. ***Left****: Known unsafe set in xyz space is displayed in red. Position components of demonstrations are overlaid.* ***Right****: Unknown unsafe set in angular velocity space is displayed in gray. Angular velocity components of demonstrations are overlaid.*

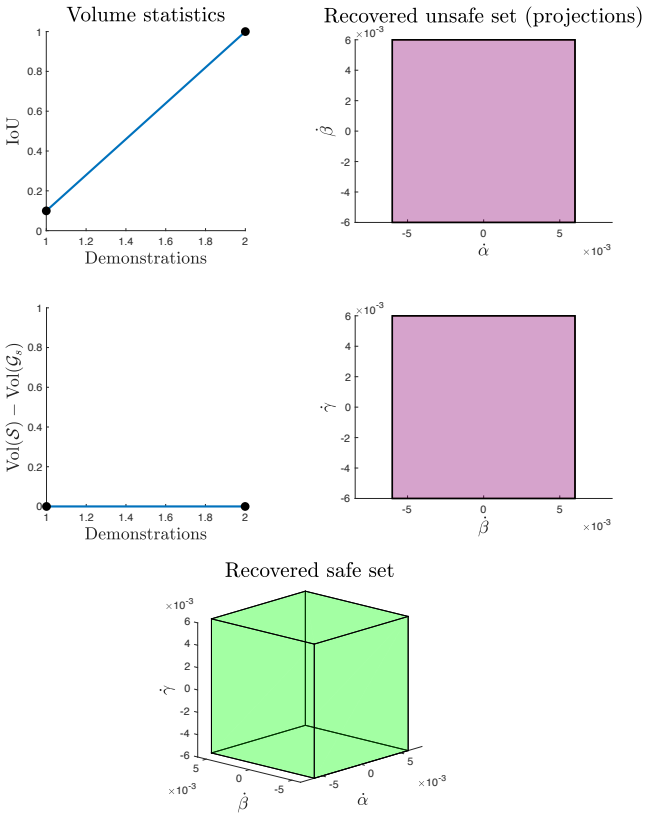

Figure 18. *Constraint recovery for 12D quadrotor, optimal demonstrations.* ***Left, top****: Unsafe set error: error in volume between the* $\hat{\mathcal{G}}^c_{\neg s}$ *and* $\mathcal{A}^c$ *as a function of demonstrations.* ***Left, bottom****: Safe set error: error in volume between* $\mathcal{G}_s$ *and* $\mathcal{S}$*.* ***Center****: projections of* $\mathcal{G}_{\neg s}$ *using all demonstrations.* ***Right****:* $\mathcal{G}_s$ *using all demonstrations.*

For discrete time systems, this distance is the maximum distance the system can travel in one time-step (see Theorem 1); for continuous systems, without time discretization, only the boundary of the unsafe set can be learned guaranteed unsafe (see Theorem 1). This is reflected in the first row of Figure 8, where cells further from the boundary of the unsafe set are not learned guaranteed unsafe. However, when leveraging a constraint parameterization, the set of constraint

states that can be learned guaranteed unsafe expands to states which can be implied unsafe by states within some distance to the boundary and the parameterization (Theorem 4). This can be seen in all the high-dimensional examples (Sections 7.2.1-7.3), where states deep in the unsafe set can be learned guaranteed unsafe.

Learnability of safe states: When gridding the constraint space, due to the independence of grid cells (i.e. learning that some cell is guaranteed unsafe can never imply that a different cell is guaranteed safe), the only cells that can be learned guaranteed safe are those visited by demonstrations. However, when using a parameterization, learning that certain states are unsafe can imply that other states must be safe, under the assumption that the true constraint can be represented with the given parametrization (see Figure 3 and the examples in Sections 7.2.1-7.3).

Conservativeness of guaranteed learned unsafe states: When gridding the constraint space, under assumptions on alignment of the grid with the unsafe set and discretization frequency, the set of guaranteed learned unsafe states is conservative (see Theorems 2 and 3). This is demonstrated in the results (Figure 8, rows 1 and 2). When these assumptions do not hold, the set of guaranteed learned unsafe states is contained within a padded version of the true unsafe set (see Figure 8, row 3, and Figure 9 for examples where overapproximation occurs due to the time-discretization chosen). When using a parameterization, the set of guaranteed learned unsafe states is conservative for discrete time systems (see Theorem 5) and is conservative within a padded version of the true unsafe set for continuous time systems (see Corollary 4). Examples of this conservativeness are shown in Figures 13 and 18, and an example where overapproximation occurs due to continuous dynamics is shown in Figure 15.

Limitations: Some limitations of our method are as follows:

- Sampling lower-cost trajectories can be slow for systems where the set of lower-cost trajectories satisfying the known constraints, $\mathcal{T}_{\mathcal{A}}^{\xi_{xu}^*}$, is "thin". For these cases, hit-and-run sampling can be forced to take very small steps at each iteration, reducing the spread of samples inside $\mathcal{T}_{\mathcal{A}}^{\xi_{xu}^*}$. This tends to happen when the dynamics are highly constrained. In future work, we will investigate more efficient sampling techniques for when $\mathcal{T}_{\mathcal{A}}^{\xi_{xu}^*}$ takes a specific form.
- To scale to high-dimensional constraint spaces, we assume a known, relatively simple parameterization, which is in general not the case for real constraints. This has been partly addressed in Chou et al. (2019) by approximating complex unsafe sets with unions of simple unsafe sets as building blocks.
- While we want the set of guaranteed learned (un)safe states to be a conservative estimate, the level of conservativeness may be high. Excessive conservativeness can be mitigated for the parametric case by obtaining the set of constraint parameters which are consistent with the demonstrations and computing a probabilistic measure of how (un)safe a given state is based on how many consistent parameters mark it as (un)safe, as is proposed in Chou et al. (2020b).
- While our method is resilient to suboptimal demonstrations within a known bound of the globally-optimal cost, it lacks guarantees for locally-optimal demonstrations. An alternative approach using the Karush-Kahn-Tucker optimality conditions has been recently proposed in Chou et al. (2020a) to enable constraint learning from locally-optimal demonstrations. Another method Knuth et al. (2020) has been developed to handle demonstrations with large suboptimality bound, where the suboptimality arises from visual occlusions that limit the demonstrators' knowledge about the environment and thus their ability to plan optimally.

9 Conclusion

In this paper we propose an algorithm that learns constraints from demonstrations, which acts as a complementary method to IOC/IRL algorithms. We analyze the properties of our algorithm as well as the theoretical limits of what subset of a safe set and an unsafe set can be learned from only safe demonstrations. The method works well on a variety of high-dimensional system dynamics and can be adapted to work with suboptimal demonstrations. We develop two variants of our algorithm to learn constraints with various amounts of structure: a gridded version which assumes no constraint structure but scales exponentially with constraint dimension, and a parametric version which assumes known parametric constraint structure and scales gracefully to high dimensional constraint spaces. We further show that our method can also learn constraints in a feature space. Future work involves using learned constraints for probabilistically safe planning and developing safe exploration strategies for reducing the uncertainty in the learned constraint.

Acknowledgements

This work was supported in part by a Rackham first-year graduate fellowship, ONR grants N00014-18-1-2501 and N00014-17-1-2050, and NSF grants CNS-1446298, ECCS-1553873, and IIS-1750489.

References

Abbasi-Yadkori, Y., Bartlett, P. L., Gabillon, V. and Malek, A. (2017), Hit-and-run for sampling and planning in non-convex spaces, *in* 'International Conference on Artificial Intelligence and Statistics'.

Abbeel, P. and Ng, A. Y. (2004), Apprenticeship learning via inverse reinforcement learning, *in* 'International Conference on Machine Learning'.

Akametalu, A., Fisac, J., Gillula, J., Kaynama, S., Zeilinger, M. and Tomlin, C. (2014), Reachability-based safe learning with gaussian processes, *in* 'Conference on Decision and Control'.

Allaire, G., Jouve, F. and Michailidis, G. (2016), Thickness control in structural optimization via a level set method, *Struct. and Multidisciplinary Optimization* .
URL: *https://hal.archives-ouvertes.fr/hal-00985000*

Amin, K., Jiang, N. and Singh, S. P. (2017), Repeated inverse reinforcement learning, *in* 'Neural Information Processing Systems', pp. 1813–1822.

Argall, B., Chernova, S., Veloso, M. M. and Browning, B. (2009), A survey of robot learning from demonstration, *Robotics and Autonomous Systems* **57**(5), 469–483.

Armesto, L., Bosga, J., Ivan, V. and Vijayakumar, S. (2017), Efficient learning of constraints and generic null space policies, *in* 'International Conference on Robotics and Automation ', IEEE, pp. 1520–1526.

Calinon, S. and Billard, A. (2007), Incremental learning of gestures by imitation in a humanoid robot, *in* 'International Conference on Human Robot Interaction 2007', pp. 255–262.

Calinon, S. and Billard, A. (2008), A probabilistic programming by demonstration framework handling constraints in joint space and task space, *in* 'International Conference on Intelligent Robots and Systems'.

Chou, G., Berenson, D. and Ozay, N. (2018), Learning Constraints from Demonstrations, *Algorithmic Foundations of Robotics XIII, Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics, WAFR 2018, Mérida, Mexico, December 9-11, 2018*, pp. 228–245.

Chou, G., Ozay, N. and Berenson, D. (2019), Learning Parametric Constraints in High Dimensions from Demonstrations, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019*, pp. 1211–1230.

Chou, G., Ozay, N. and Berenson, D. (2020), Learning Constraints from Locally-Optimal Demonstrations under Cost Function Uncertainty, *IEEE Robotics and Automation Letters, RA-L* **5**(2), pp. 3682–3690.

Chou, G., Ozay, N. and Berenson, D. (2020), Uncertainty-Aware Constraint Learning for Adaptive Safe Motion Planning from Demonstrations, *4th Annual Conference on Robot Learning, CoRL 2020, Cambridge, MA, USA, November 16 - 18, 2020*.

Dacorogna, B. (2015), *Introduction to the calculus of variations*, Imp. College Press.

Englert, P., Vien, N. A. and Toussaint, M. (2017), Inverse kkt: Learning cost functions of manipulation tasks from demonstrations, *International Journal of Robotics Research* **36**(13-14), 1474–1488.

Golub, G. H. and Van Loan, C. F. (1996), *Matrix Computations (3rd Ed.)*.

Kalman, R. E. (1964), When is a linear control system optimal?, *Journal of Basic Engineering* **86**(1), 51–60.

Karaman, S. and Frazzoli, E. (2010), Incremental sampling-based algorithms for optimal motion planning, *in* 'Robotics: Science and Systems'.

Keshavarz, A., Wang, Y. and Boyd, S. P. (2011), Imputing a convex objective function, *in* 'ISIC', IEEE, pp. 613–619.

Khalil, H. (2002), *Nonlinear systems*, Prentice-Hall.

Kiatsupaibul, S., Smith, R. L. and Zabinsky, Z. B. (2011), An analysis of a variation of hit-and-run for uniform sampling from general regions, *Transactions on Modeling and Computer Simulation* .

Knepper, R. A., Mavrogiannis, C. I., Proft, J. and Liang, C. (2017), Implicit communication in a joint action, *in* 'International Conference on Human Robot Interaction', pp. 283–292.

Knuth, C., Chou, G., Ozay, N. and Berenson, D. (2020), Inferring Obstacles and Path Validity from Visibility-Constrained Demonstrations, *Algorithmic Foundations of Robotics XIV, Proceedings of the 14th Workshop on the Algorithmic Foundations of Robotics, WAFR 2020*, pp. 18–36.

Li, C. and Berenson, D. (2016), Learning object orientation constraints and guiding constraints for narrow passages from one demonstration, *in* 'International Symposium on Experimental Robotics', Springer.

Mehr, N., Horowitz, R. and Dragan, A. D. (2016), Inferring and assisting with constraints in shared autonomy, *in* '(Conference on Decision and Control)', pp. 6689–6696.

Morin, T. L. (1982), Monotonicity and the principle of optimality, *Journal of Mathematical Analysis and Applications* **88**(2), 665 – 674.

Ng, A. Y. and Russell, S. J. (2000), Algorithms for inverse reinforcement learning, *in* 'International Conference on Machine Learning '00', San Francisco, CA, USA, pp. 663–670.

Pais, A. L., Umezawa, K., Nakamura, Y. and Billard, A. (2013), Learning robot skills through motion segmentation and constraints extraction, International Conference on Human Robot Interaction.

Papadimitriou, C. H. and Steiglitz, K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ.

Pérez-D'Arpino, C. and Shah, J. A. (2017), C-LEARN: learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy, *in* 'International Conference on Robotics and Automation'.

Ratliff, N. D., Bagnell, J. A. and Zinkevich, M. (2006), Maximum margin planning, *in* 'International Conference on Machine Learning', pp. 729–736.

Russell, S. J. and Norvig, P. (2003), *Artificial Intelligence: A Modern Approach*.

Sabatino, F. (2015), Quadrotor control: modeling, nonlinearcontrol design, and simulation.

Schreiter, J., Nguyen-Tuong, D., Eberts, M., Bischoff, B., Markert, H. and Toussaint, M. (2015), Safe exploration for active learning with gaussian processes, *in* 'European Conference on Machine Learning'.

Shoukry, Y., Nuzzo, P., Sangiovanni-Vincentelli, A. L., Seshia, S. A., Pappas, G. J. and Tabuada, P. (2018), SMC: satisfiability modulo convex programming, *Proceedings of the IEEE* **106**(9), 1655–1679.

Singh, S., Lacotte, J., Majumdar, A. and Pavone, M. (2018), Risk-sensitive inverse reinforcement learning via semi- and non-parametric methods, *International Journal of Robotics Research* **37**(13-14).

Turchetta, M., Berkenkamp, F. and Krause, A. (2016), Safe exploration in finite markov decision processes with gaussian processes, *in* 'Neural Information Processing Systems', pp. 4305–4313.

Wächter, A. and Biegler, L. T. (2006), On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Math. Program.* **106**(1), 25–57.

Ye, G. and Alterovitz, R. (2011), Demonstration-guided motion planning, *in* 'International Symposium on Robotics Research'.

A Analysis

We review the most important results in this section:

- Theorem A.1 shows that all states that can be guaranteed unsafe must lie within some distance to the boundary of the unsafe set. Corollary A.1 shows that the set of guaranteed unsafe states shrinks to a subset of the boundary of the unsafe set when using a continuous demonstration directly to learn the constraint.
- Corollary A.2 shows that under assumptions on the alignment of the grid and unsafe set for the discrete time case, the guaranteed learned unsafe set is a guaranteed underapproximation of the true unsafe set.
- For continuous trajectories that are then discretized, Theorem A.3 shows us that the guaranteed unsafe set can be made to contain states on the interior of the unsafe set, but at the cost of potentially labeling states within some distance outside of the unsafe set as unsafe as well.
- Theorem 4 shows that for the parametric case, all states that can be guaranteed unsafe must be implied unsafe by the states within some distance to the boundary of the unsafe set and the parameterization.
- Theorem A.5 shows that for the discrete time case, the guaranteed safe and guaranteed unsafe sets are inner approximations of the true safe and unsafe sets, respectively. For the continuous time case, the recovered sets are inner approximations of a padded version of the true sets.

For convenience, we repeat the definitions and include some illustrations for the sake of visualization. For clarity, the numbers of the definitions, theorems, and corollaries in the appendix match with those in the main body.

A.1 Learnability

In this section, we will provide analysis on the learnability of unsafe sets, given the known constraints and cost function. Most of the analysis will be based off unsafe sets defined over the state space, i.e. $\mathcal{A} \subseteq \mathcal{X}$, but we will extend it to the feature space in Corollary A.2. If a state x can be learned to be guaranteed unsafe, then we denote that $x \in {\mathcal{G}^z_{\neg s}}^*$, where ${\mathcal{G}^z_{\neg s}}^*$ is the set of all states that can be learned guaranteed unsafe.

We begin our analysis with some notation.

Definition A.1. Signed distance. *Signed distance from point $p \in \mathbb{R}^m$ to set $\mathcal{S} \subseteq \mathbb{R}^m$, $\mathsf{sd}(p, \mathcal{S}) = -\inf_{y \in \partial\mathcal{S}} \|p - y\|$ if $p \in \mathcal{S}$; $\inf_{y \in \partial\mathcal{S}} \|p - y\|$ if $p \in \mathcal{S}^c$.*

The following theorem describes the nature of ${\mathcal{G}^z_{\neg s}}^*$:

Theorem A.1. Learnability (discrete time). *For trajectories generated by a discrete time dynamical system satisfying $\|x_{t+1} - x_t\| \leq \Delta x$ for all t, the set of learnable guaranteed unsafe states is a subset of the outermost Δx shell of the unsafe set: ${\mathcal{G}^z_{\neg s}}^* \subseteq \{x \in \mathcal{A} \mid -\Delta x \leq \mathsf{sd}(x, \mathcal{A}) \leq 0\}$.*

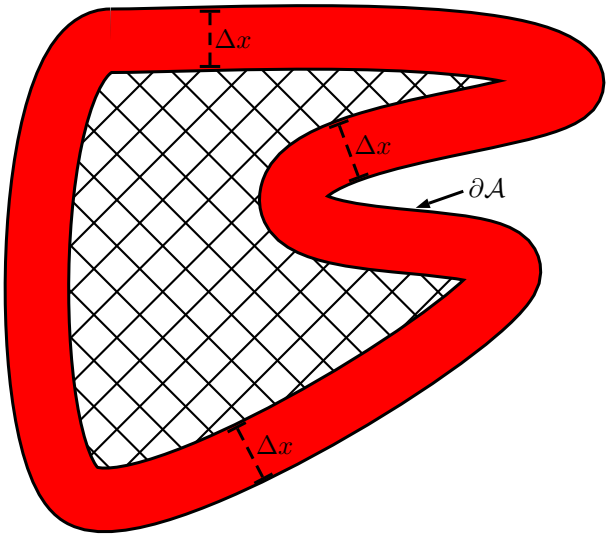

Figure A.1. *Illustration of the outermost Δx shell (shown in red) of the unsafe set $\mathcal{A}$. The hatched area cannot be learned guaranteed safe.*

Proof: Consider the case of a length T unsafe trajectory $\xi = \{x_1, \ldots, x_N\}$, $x_1 \in \mathcal{A} \vee \ldots \vee x_T \in \mathcal{A}$. For a state to be learned guaranteed unsafe, $T-1$ states in ξ must be learned safe. This implies that regardless of where that unsafe state is located in the trajectory, it must be reachable from some safe state within one time-step. This is because if multiple states in ξ differ from the original safe trajectory ξ^*, to learn that one state is unsafe with certainty means that the others should be learned safe from some other demonstration. Say that $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_T \in \mathcal{S}$, i.e. they are learned safe. Since $(\|x_{i+1} - x_i\| \leq \Delta x) \wedge (\|x_i - x_{i-1}\| \leq \Delta x)$ and $x_{i-1}, x_{i+1} \in \mathcal{S}$, x_i must be within Δx of the boundary of the unsafe set: $-\min_{y \in \partial\mathcal{A}} \|x_i - y\| \geq \Delta x$, implying $-\Delta x \leq \mathsf{sd}(x_i) \leq 0$.

□

Remark For linear dynamics, Δx can be found via

$$\underset{x \in \mathcal{X}, u \in \mathcal{U}}{\text{maximize}} \quad \|Ax + Bu - x\| \tag{17}$$

In the case of general dynamics, an upper bound on Δx can be found via

$$\Delta x \leq \sup_{x \in \mathcal{X}, u \in \mathcal{U}, t \in \{t_0, t_0+1, \ldots, T\}} \|f(x, u, t) - x\| \tag{18}$$

Corollary A.1. Learnability (continuous time). *For continuous trajectories $\xi(\cdot) : [0, T] \to \mathcal{X}$, the set of learnable guaranteed unsafe states shrinks to the boundary of the unsafe set: ${\mathcal{G}^z_{\neg s}}^* \subseteq \{x \in \mathcal{A} \mid \mathsf{sd}(x, \mathcal{A}) = 0\}$.*

Proof: The output trajectory of a continuous time system can be seen as the output of a discrete time system in the limit as the time-step is taken to 0. In this case, as long as the dynamics are locally Lipschitz continuous, $\Delta x \doteq \lim_{\Delta t \to 0} \|x(t + \Delta t) - x(t)\| \to 0$ (Khalil (2015)), and via Theorem A.1, the corollary is proved. □

Depending on the cost function, ${\mathcal{G}^z_{\neg s}}^*$ can become arbitrarily small: some cost functions are not very informative for recovering a constraint. For example, the path length cost function used in many of the experiments (which

was chosen due to its common use in the motion planning community), prevents any lower-cost sub-trajectories from being sampled from straight sub-trajectories. The overall control authority that we have on the system also impacts learnability: the more controllable the system, the more of the Δx shell is reachable. In particular, a necessary condition for any unsafe states to be learnable from a demonstration of length $T+1$ starting from x_0 and ending at x_T is for there to be more than one trajectory which steers from x_0 to x_T in $T+1$ steps while satisfying the dynamics and control constraints.

A.2 Conservativeness

For the analysis in this section, we will assume that the unsafe set has a Lipschitz boundary; informally, this means that $\partial\mathcal{A}$ can be locally described by the graph of a Lipschitz continuous function. A formal definition can be found in Dacorogna (2015). We define some notation:

Definition A.2. Normal vectors. *Denote the outward-pointing normal vector at a point $p \in \partial\mathcal{A}$ as $\hat{n}(p)$. Furthermore, at non-differentiable points on $\partial\mathcal{A}$, $\hat{n}(p)$ is replaced by the set of normal vectors for the sub-gradient of the Lipschitz function describing $\partial\mathcal{A}$ at that point (Allaire et al. (2016)).*

Definition A.3. γ-offset padding. *Define the γ-offset padding $\partial\mathcal{A}_\gamma$ as: $\partial\mathcal{A}_\gamma = \{x \in \mathcal{X} \mid x = y + d\hat{n}(y), d \in [0,\gamma], y \in \partial\mathcal{A}\}$.*

Definition A.4. γ-padded set. *We define the γ-padded set of the unsafe set $\mathcal{A}$, $\mathcal{A}(\gamma)$, as the union of the γ-offset padding and $\mathcal{A}$: $\mathcal{A}(\gamma) \doteq \partial\mathcal{A}_\gamma \cup \mathcal{A}$.*

Definition A.5. Maximum grid size. *Let $R(z_i)$ be the radius of the smallest ball which contains grid cell z_i: $R(z_i) \doteq \min_r \min_{x_i} r$, subject to $z_i \subseteq B_r(x_i)$, for some optimal center x_i.*

Furthermore, let R^* be the radius of the smallest ball which contains each grid cell $z_i, i = 1,\dots,G$: $R^* = \max(R(z_1),\dots,R(z_G))$.

We introduce the following assumption, which is illustrated in Figure A.3 for clarity:

Assumption 1: The unsafe set $\mathcal{A}$ is aligned with the grid (i.e. there does not exist a grid cell z containing both safe and unsafe states in its interior).

Theorem A.2. Discrete time conservative recovery of unsafe set. *For a discrete-time system, if Assumption 1 holds, $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}$. If Assumption 1 does not hold, then $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}(R^*)$.*

Proof: In discrete-time, we know that each trajectory sampled using Algorithm 1 starting from an optimal demonstration contains at least one truly unsafe state, i.e. for all $\xi_j, j \in \{1,\dots,N_{\neg_s}\}$, there exists $x \in \xi_j, x \in \mathcal{A}$. Then, if Assumption 1 holds, enforcing $z_i \ni x$ to be unsafe can never also enforce that some safe state $y \in \mathcal{S}$ is unsafe. If Assumption 1 does not hold, suppose that there exists $x \in \partial\mathcal{A}$ which is learned guaranteed unsafe, and that $x \in z_i$, where $(z_i \cap \mathcal{A}) \subseteq \partial\mathcal{A}$ (i.e. the grid cell only touches the boundary of the unsafe set). Then, $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}(R(z_i)) \subseteq \mathcal{A}(R^*)$. □

Note that if we deal with continuous trajectories directly, the guaranteed learnable set shrinks to a subset of the boundary of the unsafe set, $\partial\mathcal{A}$. However, if we discretize these trajectories, we can learn unsafe states lying in the interior, at the cost of conservativeness guarantees holding only for a padded unsafe set.

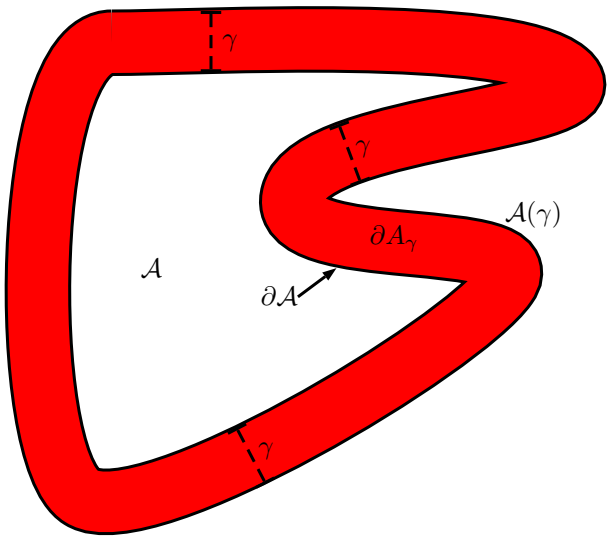

Figure A.2. *Illustration of the γ-padded set $\mathcal{A}(\gamma)$, which is the union of the red and white regions. The γ-offset padding is displayed in red. The original set $\mathcal{A}$ is shown in white.*

The following results hold for continuous time trajectories. We begin the discussion with an intermediate result we will need for Theorem A.3:

Lemma A.1. Maximum distance. *Consider a continuous time trajectory $\xi : [0,T] \to \mathcal{X}$. Suppose it is known that in some time interval $[a,b], a \le b, a, b \in [0,T]$, ξ is unsafe; denote this sub-segment as $\xi([a,b])$. Consider any $t \in [a,b]$. Then, the signed distance from $\xi(t)$ to the unsafe set, $\mathsf{sd}(\xi(t),\mathcal{A})$, is bounded by $D_\xi([a,b]) \doteq \sup_{t_1\in[a,b],t_2\in[t_1,b]} \|\xi(t_1) - \xi(t_2)\|_2$.*

Proof: Since there exists $\tilde{t} \in [a,b]$ such that $\xi(\tilde{t}) \in \mathcal{A}$, $\sup_{t\in[a,b]} \mathsf{sd}(\xi(t),\mathcal{A}) = \sup_{t\in[a,b]} \mathsf{sd}(\xi(t),\xi(\tilde{t})) \le \sup_{t_1\in[a,b],t_2\in[t_1,b]} \|\xi(t_1) - \xi(t_2)\|_2$. □

We introduce another assumption, which is illustrated in Figure A.4 for clarity:

Assumption 2: The time discretization of the unsafe trajectory $\xi : [0,T] \to \mathcal{X}$, $\{t_1,\dots,t_N\}, t_i \in [0,T]$, for all i, is chosen such that there exists at least one discretization point in the interior of each cell that the continuous trajectory passes through (i.e. if $\exists t \in [0,T]$ such that $\xi(t) \in z$, then $\exists t_i \in \{t_1,\dots,t_N\}$ such that $\xi(t_i) \in z$).

We also introduce a convention for tie-breaking in Problems 2, 4, and 5. Suppose there exists an unsafe trajectory ξ for which a safe cell z is incorrectly learned guaranteed unsafe due to time discretization. If a demonstration is added to the optimization problem which marks cell z as safe, to avoid infeasibility, we remove the unsafe trajectory ξ from the optimization problem.

Theorem A.3. Continuous-to-discrete time conservativeness. *The following results hold for continuous time systems:*

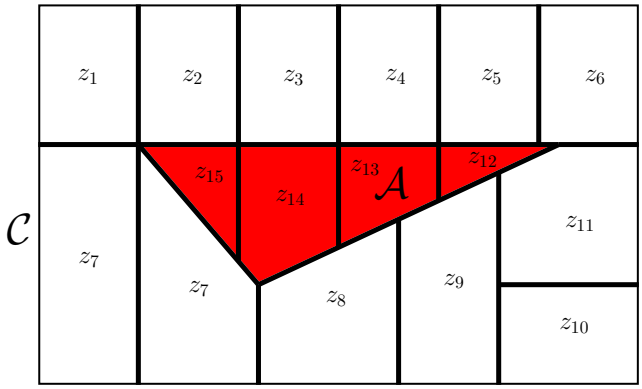

Figure A.3. *Illustration of Assumption 1 - all grid cells are either fully contained by $\mathcal{A}$ or $\mathcal{A}^c$.*

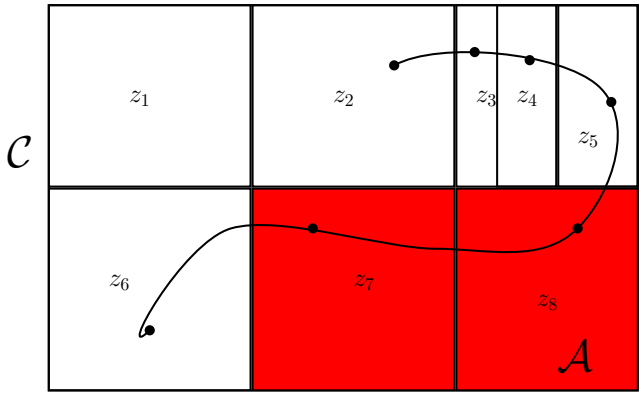

Figure A.4. *Illustration of Assumption 2: each cell z that the trajectory passes through must have a time discretization point (shown as a dot).*

1. *Suppose that both Assumptions 1 and 2 hold. Then, the learned guaranteed unsafe set $\mathcal{G}^z_{\neg_s}$, defined in Section 4.4.1, is contained within the true unsafe set $\mathcal{A}$.*

2. *Suppose that only Assumption 2 holds. Then, the learned guaranteed unsafe set $\mathcal{G}^z_{\neg_s}$ is contained within the R^*-padded unsafe set, $\mathcal{A}(R^*)$.*

3. *Suppose that neither Assumption 1 nor Assumption 2 holds. Furthermore, suppose that Problems 2, 4, and 5 are using M sub-trajectories sampled with Algorithm 1 as unsafe trajectories, and that each sub-trajectory is defined over the time interval $[a_i, b_i], i = 1, \dots, M$. Denote $D_\xi([a,b]) \doteq \sup_{t_1 \in [a,b], t_2 \in [t_1,b]} \|\xi(t_1) - \xi(t_2)\|_2$, for some trajectory ξ. Denote $D^* \doteq \max_{i \in \{1,\dots,M\}} D^*_{\xi_i}([a_i, b_i])$. Then, the learned guaranteed unsafe set $\mathcal{G}^z_{\neg_s}$ is contained within the $D^* + R^*$-padded unsafe set, $\mathcal{A}(D^* + R^*)$.*

Proof: Let's prove the case where both Assumptions 1 and 2 hold. By Assumption 1, all cells z which contain unsafe states $x \in \mathcal{A}$ must be fully contained in the unsafe set: $z \in \mathcal{A}$. Now, suppose there exists a trajectory $\xi : [0, T] \to \mathcal{X}$ which is unsafe (i.e it satisfies the known constraints and has lower cost than a demonstration). Then, there exists at least one $t \in [0, T]$ such that $\xi(t) \in \mathcal{A}$. By Assumption 2, there exists a discretization point $t_i \in [0, T]$ such that $\xi(t_i)$ lies within some cell z, and $z \in \mathcal{A}$ by Assumption 1. Hence, we will only learn grid cells within $\mathcal{A}$ to be unsafe: $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}$.

If only Assumption 2 holds, $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}(R^*)$ due to the gridding: suppose there exists a cell z_k containing both safe and unsafe states which is learned guaranteed unsafe. Then, by padding the unsafe set to contain any grid cell $z_1, \dots, z_G$, z_k is fully contained, and hence the algorithm returns a conservative estimate of the $D^* + R_k \leq D^* + R^*$-padded unsafe set.

Let's prove the case where neither assumption holds. Suppose in this case, there exists a cell $z \not\subseteq \mathcal{A}$ which is truly safe, but for which we have no demonstration that says cell z is safe. Now, suppose there exists an unsafe trajectory $\xi_j([a_j, b_j])$ passing through z which violates Assumption 2. Suppose that $\xi_j(t_i) \in z$, and $\{t_1, \dots, t_N\}$ is chosen such that for all $j \in \{1, \dots, N\} \setminus \{i\}$, $\xi_j(t_i)$ belongs to a known safe cell. Then, we may incorrectly learn that $z \in \mathcal{G}^z_{\neg_s}$, as we force at least one point in the sampled trajectory to be unsafe. Via Lemma A.1, we know that $\xi_j(t_i)$ is at most $D_{\xi_j}([a_j, b_j])$ signed distance away from $\mathcal{A}$. Hence, for this trajectory, any learned guaranteed unsafe state must be contained in the $D_{\xi_j}([a_j, b_j])$-padded unsafe set. For this to hold for all unsafe trajectories sampled with Algorithm 1, we must pad the unsafe set by D^*. Lastly, to account for the gridding, suppose that $\xi_{j^*}(t_i)$ is contained in cell z_k, which is then marked unsafe. Then, by padding the set to contain z_k, the algorithm returns a conservative estimate of the $D^* + R_k \leq D^* + R^*$-padded unsafe set.

□

Remark In practice, we observe that the bound in Theorem A.3 when using only Assumption 1 is quite conservative, and as more demonstrations are added to the optimization, using the tie-breaking rule described previously removes the overapproximations described by Theorem A.3. Furthermore, though the experiments are implemented using only Assumption 1, ensuring Assumption 2 also holds is straightforward as long as the grid cells are large enough such that finding a sufficiently fine time-discretization is efficient.

Remark Note that for the cases where Assumption 1 does not hold, safe states can be incorrectly forced to be unsafe; thus, the constraint recovery program can become infeasible. In these situations, we use the tie-breaking rule described before the statement of Theorem A.3 to keep the program feasible.

Remark If some state x on a demonstration lies directly on the boundary between two grid cells z_i and z_j, neither z_i nor z_j is enforced to be safe unless either of z_i or z_j is learned to be unsafe; then the other grid cell can be labeled safe. Furthermore, the demonstrations that appear to lie on the boundary of the unsafe set actually lie in the interior of the safe set and very close to the boundary due to the solvers' numerical tolerance; hence we do not actually have any demonstrations lying exactly on the boundary of any grid cells in the experiments.

Corollary A.2. Continuous-to-discrete feature space conservativeness. *Let the feature mapping $\phi(x)$ from the state space to the constraint space be Lipschitz continuous with Lipschitz constant L. Then, the following results hold:*

1. *Suppose both Assumptions 1 and 2 (used in Theorem 3) hold. Then, our method ensures* $\mathcal{G}^z_{\neg_s} \subseteq \mathcal{A}$.
2. *Suppose only Assumption 2 holds. Then, our method recovers a guaranteed subset of the* LR^**-padded unsafe set,* $\mathcal{A}(LR^*)$*, in the feature space.*
3. *Suppose neither Assumption 1 nor Assumption 2 holds. Then, our method recovers a guaranteed subset of the* $L(D^* + R^*)$*-padded unsafe set,* $\mathcal{A}(L(D^* + R^*))$*, where* D^* *is as defined in Theorem 3.*

Proof: Under Assumptions 1 and 2, the result follows directly from the logic in Theorem A.3. Now, consider the case where only Assumption 2 holds. From the definition of Lipschitz continuity, $\|\phi(x) - \phi(y)\| \leq L\|x - y\|$. From Theorem A.3, the unsafe set estimate is a subset of the R^*-padded estimate in the continuous space case. Using Lipschitz continuity, the value of the feature can at most change by LR^* from the boundary of the true constraint set to the boundary of the padded set; hence, the statement holds. Analogous reasoning holds for the case where neither assumption holds. □

A.3 Learnability: Parametric

In this section, we develop results for learnability of the unsafe set in the parametric case. We begin with the following notation:

Definition A.6. Implied unsafe set. *For some set* $\mathcal{B} \subseteq \Theta$*, denote*

$$I(\mathcal{B}) \doteq \bigcap_{\theta \in \mathcal{B}} \{x \mid g(x, \theta) \leq 0\} \tag{19}$$

as the set of states that are implied unsafe by restricting the parameter set to $\mathcal{B}$*. In words,* $I(\mathcal{B})$ *is the set of states for which all* $\theta \in \mathcal{B}$ *mark as unsafe.*

Lemma A.2. *Suppose* $\mathcal{B} \subseteq \hat{\mathcal{B}}$*, for some other set* $\hat{\mathcal{B}}$*. Then,* $I(\hat{\mathcal{B}}) \subseteq I(\mathcal{B})$.

Proof: By definition,

$$\begin{aligned} I(\hat{\mathcal{B}}) &= \bigcap_{\theta \in \hat{\mathcal{B}}} \{x \mid g(x, \theta) \leq 0\} \\ &= \bigcap_{\theta \in \left(\mathcal{B} \cup (\hat{\mathcal{B}} \setminus \mathcal{B})\right)} \{x \mid g(x, \theta) \leq 0\} \\ &\subseteq \bigcap_{\theta \in \mathcal{B}} \{x \mid g(x, \theta) \leq 0\} \\ &= I(\mathcal{B}). \end{aligned}$$

□

Lemma A.3. *Denote the* Δx*-shell of* $\mathcal{A}$ *as* $\mathcal{A}_{\Delta x}$*, where* Δx *is as defined in Theorem A.1. Then, each unsafe trajectory* ξ_j *with start and goal states in the safe set contains at least one state in* $\mathcal{A}_{\Delta x}$*:* $\forall j \in \{1, \ldots, N_{\neg_s}\}, \exists x \in \xi_j, x \in \mathcal{A}_{\Delta x}$.

Proof: For each unsafe trajectory ξ_j with start and goal states in the safe set, there exists $x \in \xi_j, x \in \mathcal{A}$. Further, if there exists $x \in \xi_j \in (\mathcal{A} \setminus \mathcal{A}_{\Delta x})$, then there also exists $x \in \xi_j \in \mathcal{A}_{\Delta x}$. For contradiction, suppose there exists a time $\hat{t} \in \{1, \ldots, T_j\}$ for which $\xi_j(\hat{t}) \in (\mathcal{A} \setminus \mathcal{A}_{\Delta x})$ and $\nexists t \in \{1, \ldots, T_j\}$ for which $\xi_j(t) \in \mathcal{A}_{\Delta x}$. But this implies $\exists t < \hat{t}, \|\xi(t) - \xi(t+1)\| > \Delta x$ or $\exists t > \hat{t}, \|\xi(t) - \xi(t-1)\| > \Delta x$, i.e. to skip deeper than Δx into the unsafe set without first entering the Δx shell, the state must have changed by more than Δx in a single time-step. Contradiction. An analogous argument holds for the continuous-time case. □

Denote as $\mathcal{G}^*_{\neg_s}$ the learnable set of unsafe states. Further denote as $\mathcal{F}_{\Delta x}$ the set of parameters that sets all states in $\mathcal{A}_{\Delta x}$ as unsafe and all states on safe trajectories as safe. Last, denote as $I(\mathcal{F}_{\Delta x})$ the set of states that are implied as unsafe by restricting the parameter set to $\mathcal{F}_{\Delta x}$. The following result states that in discrete time, $\mathcal{G}^*_{\neg_s}$ is contained by $I(\mathcal{F}_{\Delta x})$. Furthermore, in continuous time, the same holds, except the Δx shell is replaced by the boundary of the unsafe set, $\partial\mathcal{A}$.

Theorem A.4. Discrete time learnability for parametric constraints. *For trajectories generated by discrete time systems,* $\mathcal{G}_{\neg_s} \subseteq \mathcal{G}^*_{\neg_s} \subseteq I(\mathcal{F}_{\Delta x})$*, where*

$$\begin{aligned} \mathcal{F}_{\Delta x} = \{\theta \mid \forall i \in \{1, \ldots, N_s\}, \forall x \in \xi^*_i, g(x, \theta) > 0, \\ \forall x \in \mathcal{A}_{\Delta x}, g(x, \theta) \leq 0\} \end{aligned}$$

Proof: Recall that $\mathcal{G}_{\neg_s} \doteq \bigcap_{\theta \in \mathcal{F}} \{x \mid g(x, \theta) \leq 0\}$, where $\mathcal{F}$ is the feasible set of Problem 3:

$$\begin{aligned} \mathcal{F} = \{\theta \mid \forall i \in \{1, \ldots, N_s\}, \forall x \in \xi^*_i, g(x, \theta) > 0, \\ \forall j \in \{1, \ldots, N_{\neg_s}\}, \exists x \in \xi_j, g(x, \theta) \leq 0\} \end{aligned}$$

We can then show that $\mathcal{F}_{\Delta x} \subseteq \mathcal{F}$, since enforcing that $g(x, \theta) \leq 0$ for all $x \in \mathcal{A}_{\Delta x}$ implies that there exists $x \in \xi_j$, for all $j \in \{1, \ldots, N_{\neg_s}\}$ such that $g(x, \theta) \leq 0$, via Lemma A.3. Then, via Lemma A.2, $\mathcal{G}_{\neg_s} = I(\mathcal{F}) \subseteq I(\mathcal{F}_{\Delta x})$. As this holds for any arbitrary set of trajectories, $\mathcal{G}^*_{\neg_s} \subseteq I(\mathcal{F}_{\Delta x})$ as well, and $\mathcal{G}_{\neg_s} \subseteq \mathcal{G}^*_{\neg_s}$. □

Corollary A.3. Continuous-time learnability for parametric constraints. *For trajectories generated by continuous time systems,* $\mathcal{G}_{\neg_s} \subseteq \mathcal{G}^*_{\neg_s} \subseteq I(\mathcal{F}_{\partial\mathcal{A}})$*, where*

$$\begin{aligned} \mathcal{F}_{\partial\mathcal{A}} = \{\theta \mid \forall x \in \xi^*_i, \forall i \in \{1, \ldots, N_s\}, g(x, \theta) > 0, \\ \forall x \in \partial\mathcal{A}, g(x, \theta) \leq 0\} \end{aligned}$$

Proof: Since going from discrete time to continuous time implies $\Delta x \to 0$, $\mathcal{A}_{\Delta x} \to \partial\mathcal{A}$. Then, the logic from the proof of Theorem A.4 can be similarly applied to show the result. □

A.4 Conservativeness: Parametric

We write conditions for conservative recovery of the unsafe set and safe set when solving Problems 3 and 6 for discrete time and continuous time systems.

Theorem A.5. *For a discrete-time system, if* M *in Problem 6 is chosen to be greater than* $\max(M_1, M_2)$*, where* $M_1 = \max_{x_i \in \xi_s} \max_\theta \max_j (H(\theta)x_i - h(\theta))_j$ *and* $M_2 = \max_{x_i \in \xi_{\neg_s}} \max_\theta \max_j (H(\theta)x_i - h(\theta))_j$*,* $\mathcal{G}_{\neg_s} \subseteq \mathcal{A}$ *and* $\mathcal{G}_s \subseteq \mathcal{S}$.

Proof:

We first prove that $\mathcal{G}_{\neg_s} \subseteq \mathcal{A}$. Consider first the case where $M = \infty$ and therefore Problem 6 exactly enforces that at least one state in each unsafe trajectory is unsafe and all states on demonstrations are safe.

Suppose for contradiction that there exists some $x \in \mathcal{G}_{\neg_s}, x \notin \mathcal{A}$. By definition of $\mathcal{G}_{\neg_s}$, $g(x, \theta) \leq 0$, for all $\theta \in \mathcal{F}$,

where $\mathcal{F}$ is the feasible set of parameters θ in Problem 3. However, as $x \notin \mathcal{A}$, but for all $\theta \in \mathcal{F}, g(x,\theta) \leq 0$ we know that $\theta_{\mathcal{A}} \notin \mathcal{F}$, where $\theta_{\mathcal{A}}$ is the parameter associated with the true unsafe set $\mathcal{A}$. However, $\mathcal{F}$ will always contain $\theta_{\mathcal{A}}$, since:

- $\theta_{\mathcal{A}}$ satisfies $g(x,\theta_{\mathcal{A}}) > 0$ for all x in safe demonstrations, since all demonstrations are safe with respect to the true $\theta_{\mathcal{A}}$.
- For each trajectory $\xi_{\neg s}$ sampled using Algorithm 1, there exists $x \in \xi_{\neg s}$ such that $g(x,\theta_{\mathcal{A}}) \leq 0$.

We come to a contradiction, and hence for $M = \infty$, $\mathcal{G}_{\neg s} \subseteq \mathcal{A}$.

Now, we consider the conditions on M such that choosing $M \geq$ const or $M = \infty$ causes no changes in the solution of Problem 6. M must be chosen such that 1) $H(\theta)x_i - h(\theta) > -M\mathbf{1} \Leftrightarrow H(\theta)x_i - h(\theta) > -\infty\mathbf{1}$, for all safe states $x_i \in \xi_s$, and 2) $H(\theta)x_i - h(\theta) \leq M\mathbf{1} \Leftrightarrow H(\theta)x_i - h(\theta) \leq M\mathbf{1}$ for all states x_i on unsafe trajectories $\xi_{\neg s}$. Condition 1 is met if $-M < \min_{x_i \in \xi_s} \min_\theta \min_j (H(\theta)x_i - h(\theta))_j$, where v_j denotes the j-th element of vector v; denote as M_1 an M which satisfies this inequality. Condition 2 is met if $M \geq \max_{x_i \in \xi_{\neg s}} \max_\theta \max_j (H(\theta)x_i - h(\theta))_j$; denote as M_2 an M which satisfies this inequality. Then, M should be chosen to satisfy $M > \max(M_1, M_2)$.

The proof that $\mathcal{G}_s \subseteq \mathcal{S}$ is analogous. If there exists $x \in \mathcal{G}_s, x \notin \mathcal{S}$, $g(x,\theta) > 0$, for all $\theta \in \mathcal{F}$, then $\theta_{\mathcal{A}} \notin \mathcal{F}$. We follow the same reasoning from before to show that $\theta_{\mathcal{A}} \in \mathcal{F}$ for $M = \infty$. Now, provided the condition on M holds, we reach a contradiction. □

Corollary A.4. *For a continuous-time system, where demonstrations are time-discretized as discussed in Section A.2, if M is chosen as in Theorem A.5, $\mathcal{G}_s \subseteq \mathcal{S}$ and $\mathcal{G}_{\neg s} \subseteq \mathcal{A}(D^*)$, where D^* is as defined in Theorem A.3.*

Proof: The reasoning for $\mathcal{G}_s \subseteq \mathcal{S}$ follows from the proof of Theorem A.5.

For proving $\mathcal{G}_{\neg s} \subseteq \mathcal{A}(D^*)$, we follow the proof of Theorem A.3 until it is shown that any learned guaranteed unsafe state must be contained in the $\mathcal{A}(D^*)$. However, for the parametric case, there is no notion of a grid and hence the further padding by R^* is unnecessary. □