Active Learning for Estimating Reachable Sets for Systems with Unknown Dynamics

Chakrabarty, Ankush; Raghunathan, Arvind; Di Cairano, Stefano; Danielson, Claus

TR2020-164 December 16, 2020

Abstract

This paper presents a data-driven method for computing reachable sets where active learning is used to reduce the computational burden. Set-based methods used to estimate reachable sets typically do not scale well with state-space dimension, or rely heavily on the existence of a model. If such a model is not available, it is simple to generate state trajectory data by numerically simulating black-box oracles of systems (whose dynamics are unknown) from sampled initial conditions. Using these data samples, the estimation of reachable sets can be posed as a classification problem, wherein active learning (AL) can intelligently select samples that are most informative and least similar to previously labeled samples. By exploiting submodularity, the actively learned samples can be selected efficiently, with bounded sub-optimality. Our proposed framework is illustrated by estimating the domains of attractions of model predictive controllers (MPCs) and reinforcement learners. We also consider a scenario where there are two oracles that differ with respect to evaluation costs and labeling accuracy. We propose a framework to reduce the dependency of the expensive oracle in labeling samples using disagreement-based active learning (DBAL). The potential of the DBAL algorithm is demonstrated on a solver selection problem for real-time MPC

IEEE Transactions on Cybernetics

Active Learning for Estimating Reachable Sets for Systems with Unknown Dynamics

Ankush Chakrabarty[†], Claus Danielson, Stefano Di Cairano, Arvind Raghunathan[⋆]

***Abstract*—This paper presents a data-driven method for computing reachable sets where active learning is used to reduce the computational burden. Set-based methods used to estimate reachable sets typically do not scale well with state-space dimension, or rely heavily on the existence of a model. If such a model is not available, it is simple to generate state trajectory data by numerically simulating black-box oracles of systems (whose dynamics are unknown) from sampled initial conditions. Using these data samples, the estimation of reachable sets can be posed as a classification problem, wherein active learning (AL) can intelligently select samples that are most informative and least similar to previously labeled samples. By exploiting submodularity, the actively learned samples can be selected efficiently, with bounded sub-optimality. Our proposed framework is illustrated by estimating the domains of attractions of model predictive controllers (MPCs) and reinforcement learners. We also consider a scenario where there are two oracles that differ with respect to evaluation costs and labeling accuracy. We propose a framework to reduce the dependency of the expensive oracle in labeling samples using disagreement-based active learning (DBAL). The potential of the DBAL algorithm is demonstrated on a solver selection problem for real-time MPC.**

***Index Terms*—Machine learning; safe sets; submodularity; imbalanced learning; design of experiments; domain of attraction; nonlinear systems; invariant sets; reinforcement learning.**

I. Introduction

Estimating sets such as domains of attraction, reachable, and invariant sets for nonlinear dynamical systems in a computationally efficient manner is a fundamental challenge in stability analysis [1] and control design [2]–[6].

This set estimation problem is usually tackled by generating a local Lyapunov function, whose level set demarcates the boundary of a domain of attraction. Consequently, multiple algorithms have been proposed to construct Lyapunov functions numerically [7]–[9], using a model of the underlying system dynamics. An approach that has been recently used is based on sum-of-squares programming [10]–[12], wherein a rational Lyapunov function and a polynomial static output control law is generated to estimate and manipulate a reachable set for polynomial systems by solving quasi-convex bilinear matrix inequalities. When the underlying dynamics are non-polynomial, a couple of recent papers have employed truncated Taylor expansions and Chebyshev fitting to tackle the estimation problem [13], [14]. Another approach that is currently being studied is based on using a linear fractional transformation to construct a differential-algebraic system which simplifies the stability analysis by reducing the set estimation problem using maximal annihilators [15], [16].

As opposed to the method developed here, all the above methods require a closed-form analytical model of the system dynamics, which may not be readily available or simple to obtain if the plant dynamics are described by complex simulation models, or if the plant is in closed loop with a numerical control algorithm, such as model predictive control. For systems that do not conform to structured representations or when closed-form causal models are not readily available (for example, when the dynamics are represented by complex blocks in Simulink/Labview/Modelica, represent approximations of partial differential equations, or contain differential algebraic equations such as in fluid flows, power flow, or industrial automation), the determination of reachable sets is near impossible using analytical methods. In lieu of closed-form models, numerical simulations offer a fast and scalable alternative to collecting reachability or invariance information. Sampling-based approaches, based on direct simulations, enable the reachable set estimation problem to be cast as a classification problem [17]–[19]. Data-driven methods such as these eliminate the need to exploit structure of specific nonlinearities, enabling implementation on a wide range of dynamical systems. A limitation of these methods is that accurate estimation of reachable sets is only ensured with an infinite number of samples, which is impractical.

A more practical sampling method, referred to as active learning (AL), which in different contexts may be also referred to as 'optimal experiment design' or 'directed sampling', hinges on selecting the most informative samples within the state space. In this paper, we query the oracle batch-wise from a pool of unlabeled samples: this is referred to as 'pool-based batch selection' [20] and is generally faster than single-instance selection methods. A common theme amongst pool-based batch selection AL algorithms is that samples are chosen according to metrics that increase the information contained and reduce overlap with prior labeled samples [21]–[24]. Often times, these methods cannot rigorously certify the quality of the solutions obtained.

In this paper, we exploit submodularity in our proposed pool-based batch selection AL procedure to ensure near-optimality of solutions [25]. Although we are not the first to propose the utilization of submodularity in this setting, see for example [26], [27], our proposed approach does not require specific learners and does not involve inverting large Fisher information matrices. Some preliminary results using submodularity and active learning for reachable set estimation

[†]Corresponding author: Ankush Chakrabarty. Phone: +1 (617) 758-6175. Email: chakrabarty@merl.com.

[⋆]All authors ({danielson, dicairano, raghunathan}@merl.com) are affiliated with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

have been reported in our previous work [28]. In this paper, after refining and formalizing our initial results we provide two extensions to such results. First, we consider the case when the size of the unlabeled sample set is particularly large. Having a large number of unlabeled samples could be used to ensure that the unlabeled set is well-dispersed on the admissible state-space. Such a large set also implies that active learning on such a set will prove computationally challenging. To reduce the computational expenditure involved in the greedy submodular maximization algorithm in this setting, we provide two variants based on random subsampling of the unlabeled set and partitioning the unlabeled sample set for distributed computing on multiple clusters. Our approach is related to methods for distributed implementations and stochastic variants of greedy submodular maximization that have been reported in [29]–[31] for applications such as sensor scheduling [32], [33]. Second, we consider the case when two oracles are available: a strong (infallible) oracle that is expensive to query, and a weak (fallible) oracle that is cheap to query. We use tools from disagreement-based active learning (DBAL) [34] in order to solve this problem by estimating a region where the two oracles disagree, and to use this disagreement learner to reduce the number of queries made to the strong oracle when solving the primary problem of identifying the reachable set.

In summary, the specific **contributions** of this paper are:

(i) we introduce a batch-mode active learning procedure to select relevant samples for learning reachable set boundaries in a sampling-based, data-driven manner and provide a method for obtaining near-optimal samples in polynomial time despite the NP-hard nature of the original learning problem;

(ii) providing rigorous mathematical conditions for the existence of such boundaries; and,

(iii) we demonstrate the generalizability of our proposed approach to multiple classifiers by considering an active learning problem with weak and strong labelers.

The paper is organized as follows. The motivation for the problem investigated in the paper is discussed in Section II. For the single oracle case, an active learning methodology is provided in Section III. Fast variants of the algorithm for large unlabeled sets are discussed in Section IV and an algorithm that handles the scenario when a weak and strong oracle are simultaneously available in Section V. Simulation results in Section VI demonstrate the potential of the method. We present our conclusions in Section VII.

Notation

We denote by $\mathbb{R}$ the set of real numbers, $\mathbb{R}_+$ as the set of positive reals, and $\mathbb{N}$ as the set of natural numbers. The symbol $\mathbb{E}(\cdot)$ denotes the expectation operator. For a set $\mathcal{S}$, we denote its cardinality by $|\mathcal{S}|$, its interior (if it exists) by $\operatorname{int}\mathcal{S}$, and its boundary (if the set is closed) by $\partial\mathcal{S}$. The union, intersection, difference, and symmetric difference of sets $\mathcal{A}$ and $\mathcal{B}$ are denoted $\mathcal{A}\cup\mathcal{B}$, $\mathcal{A}\cap\mathcal{B}$, $\mathcal{A}\backslash\mathcal{B}$, and $\mathcal{A}\triangle\mathcal{B}$, respectively. The empty set is denoted $\emptyset$. The power set of a set $\mathcal{A}$ is denoted $\mathbf{2}^{\mathcal{A}}$. The notation ln represents the natural logarithm. The big-O notation is represented by $\mathbf{O}(\cdot)$. A function $f(x)$ is semi-continuous at $\bar{x}$ if $\forall\epsilon > 0$ there exists $\delta > 0$ such that $f(x) \le f(\bar{x}) + \epsilon$ (upper semi-continuous) or $f(\bar{x}) - \epsilon \le f(x)$ (lower semi-continuous) for all $\|x - \bar{x}\| \le \delta$.

II. Motivation

Consider a dynamical system of the form

$$x_{t+1} = f(x_t, u_t), \tag{1a}$$

$$x_t \in \mathbb{X},\ u_t \in \mathbb{U} \tag{1b}$$

where $x_t \in \mathbb{R}^{n_x}$ and $u_t \in \mathbb{R}^{n_u}$ are the state and input, respectively, at discrete-time $t \in \mathbb{N}$. We assume that the constraint sets $\mathbb{X} \subset \mathbb{R}^{n_x}$ and $\mathbb{U} \subset \mathbb{R}^{n_u}$ are compact, convex, and contain the origin in their interiors. We further assume that the dynamics (1a) are continuous and the origin is the unique equilibrium state-input pair $f(0, 0) = 0$. Note that, although we focus on nonlinear discrete-time systems in this paper, the proposed method can be applied directly to continuous-time systems and/or linear systems.

Definition 1 (T-step Backward-Reachable Set)**.** *The T-step backward-reachable set $\mathcal{R}_T(\Omega) \subseteq \mathbb{X}$ of a compact set $\Omega \subseteq \mathbb{X}$ is the set of all initial conditions $x_0 \in \mathbb{X}$ for which there exists a sequence of inputs such that $u_t \in \mathbb{U}$ such that $x_{t+1} = f(x_t, u_t) \in \mathbb{X}$ for $t = 0, \ldots, T-1$ and $x_T \in \Omega$ where $T \in \mathbb{N}$, that is:*

$$\begin{aligned}\mathcal{R}_T(\Omega) = \big\{x_0 \in \mathbb{X} : \exists u_t \in \mathbb{U},& \\ x_{t+1} = f(x_t, u_t) \in \mathbb{X}, x_T \in \Omega\big\}&.\end{aligned} \tag{2}$$

The sets described in Definition 1 are also referred to as T-step controllable sets. A special case of Definition 1 are backward reachable sets for closed-loop systems $x_{t+1} = f(x_t, \kappa(x_t))$ with a fixed controller $u_t = \kappa(x_t)$,

$$\begin{aligned}\mathcal{R}_T(\Omega) = \big\{x_0 \in \mathbb{X} : \kappa(x_t) \in \mathbb{U},& \\ x_{t+1} = f(x_t, \kappa(x_t)) \in \mathbb{X}, x_T \in \Omega\big\}&.\end{aligned} \tag{3}$$

We assume that information about the reachable set $\mathcal{R}_T(\Omega)$ is available through an oracle $\mathcal{O} : \mathbb{X} \to \{-1, 1\}$ that evaluates the indicator function of the reachable set. With abuse of notation, we will conflate oracles $\mathcal{O} : \mathbb{X} \to \{-1, 1\}$ and the indicator function

$$\mathcal{O}(x) = \begin{cases} +1 & \text{if } x \in \mathcal{R}_T(\Omega) \\ -1 & \text{if } x \notin \mathcal{R}_T(\Omega). \end{cases} \tag{4}$$

This paper is motivated by the fact that oracles (4) are typically expensive to query. The following four examples illustrate the concept of an oracle (4).

Example 1 (Nonlinear Model Predictive Control)**.** The oracle (4) of the reachable set (2) can be evaluated by solving the following finite-horizon constrained optimal control problem,

$$\min\ p(x_T) + \sum\nolimits_{t=0}^{T-1} q(x_t, u_t) \tag{5a}$$

$$\text{subject to: } x_{t+1} = f(x_t, u_t) \text{ for } t = 0, \ldots, T-1 \tag{5b}$$

$$x_t \in \mathbb{X},\ u_t \in \mathbb{U},\ x_T \in \Omega \tag{5c}$$

where the terminal $p(\cdot)$ and stage-costs $q(\cdot,\cdot)$ are arbitrary since we are only concerned with the feasibility of the optimal control problem (5). For (5), the oracle (4) has the form

$$\mathcal{O}(x_0) = \begin{cases} +1 & \text{if (5) is feasible} \\ -1 & \text{otherwise.} \end{cases}$$

This oracle is computationally expensive to evaluate since it requires solving an optimization problem that is non-convex when the model is nonlinear. □

Example 2 (Simulation)**.** For a system with closed-loop dynamics $x_{t+1} = f(x_t, \kappa(x_t))$, the oracle (4) of the reachable set (3) can be evaluated through high-fidelity simulations. This includes models constructed using modeling software like Simulink/Modelica, models arising from finite element/difference approximations of partial differential equations, or models designed in dynamics-rendering engines like Bullet/MuJoCo [35]. The dynamics (1a) are simulated over the time interval $[0, T]$ with initial condition $x(0) = x_0$, resulting in a sequence of control inputs $u(t) = u_t$ and states $x(t) = x_t$. The oracle (4) then has the form

$$\mathcal{O}(x_0) = \begin{cases} +1 & \text{if } u_t \in \mathbb{U}, x_t \in \mathbb{X}, x_T \in \Omega \\ -1 & \text{otherwise.} \end{cases} \tag{6}$$

This oracle is computationally expensive to evaluate since it requires simulating a high-fidelity model of the dynamics (1a). □

Example 3 (Experiment)**.** Similarly to Example 2, the oracle (4) of the reachable set (3) can be evaluated for a closed-loop system through experiments with initial condition x_0. □

Example 4 (Cloud/Database Query)**.** Legacy systems that have been operating for a long period of time often have large databases that document their history of operation. This database can be queried to evaluate the oracle (6). Similarly, if there are many (nearly) identical systems sharing data through the cloud, this data serves as an oracle (6). □

In Examples 3 and 4, the oracles (6) do not require any knowledge of the dynamics (1a). Thus, an advantage of using (4) is that we can estimate the reachable set without knowing the closed-form model of the dynamics (1a). Furthermore, although the model in Example 2 is known, it can be very complex, for high accuracy, without making the problem of estimating the reachable set intractable. This is important for many applications where feasibility is determined experimentally such as in buildings [36], biology [37], or healthcare [38]. One can also use this method to validate 'black-box' controllers, where the controller's behavior is completely unknown, such as for control with deep neural networks, therein generating some trust in the closed-loop system.

Our *objective* is to estimate the reachable set of systems with complex model representations with limited oracle queries. To this end, we use active learning. Concretely, we leverage machine intelligence to iteratively select initial conditions $x_0 \in \mathbb{X}$ for which the oracle (4) should be queried for labeling. With these labeled samples (which will be the inputs to a classification algorithm), our goal is to characterize inner approximations of the reachable set for closed-loop system analysis and on-line control. Due to the iterative framework of active learning algorithms, one expects gradual improvement of the inner approximation with more samples, which is an advantage of this framework over analytical methods that generate conservative estimates of the inner approximation.

III. Active Learning with Infallible Oracle

The major advantage of AL methods over traditional supervised learning is the iterative improvement of learning performance by systematically utilizing prior learners. In most AL algorithms, one starts with a labeled set $\mathcal{L}_0$ and an unlabeled set $\mathcal{S}$ in the state space $\mathbb{X}$. The goal is to select N_s most informative samples iteratively in batches. Specifically, in the kth iteration, a set of samples $\mathcal{L}'_k \subset \mathcal{S}$ is chosen based on the prior classifier. Only the samples $\mathcal{L}'_k$ are labeled by the oracles, and the labels are added to the accumulated labeled set. Once N_s samples have been selected, the AL terminates. More details are presented in the key steps below.

We use the initial labeled set $\mathcal{L}_0$ and the unlabeled set $\mathcal{S}$ (usually obtained by gridding or sampling methods upon the state space $\mathbb{X}$, see for example, [17]) to generate a non-trivial initial classifier. Usually, we can select $\mathcal{S}$ to have large cardinality and good coverage of $\mathbb{X}$ since sampling is a cheap computational procedure compared to querying the oracle, and well-distributed samples ensure that informative samples exist near the boundary of the reachable set we wish to estimate.

A classifier $\psi_0 : \mathbb{X} \to [-1, 1]$ is trained using the features $\mathcal{L}_0$. The overall idea is that the classifier ψ_0 will actively learn a set of 'important' samples that will be appended to $\mathcal{L}_0$ to form the new training set $\mathcal{L}_1$. This will in turn induce a classifier ψ_1 and the workflow of the AL algorithm will proceed as:

$$\mathcal{L}_0 \to \psi_0 \to \mathcal{L}_1 \to \psi_1 \to \cdots \mathcal{L}_k \to \psi_k \to \mathcal{L}'_k \to \cdots .$$

At the kth iteration, a prior classifier ψ_{k-1} informs the selection of B samples within $\mathcal{S}$ that minimizes overlap compared to the current label set $\mathcal{L}_k$ while providing the most discerning information. The classifiers $\psi_k : \mathbb{X} \to [-1, 1]$ can be interpreted as the belief, based on the dataset $\mathcal{L}_k$, that a state $x \in \mathbb{X}$ belongs to the reachable set (2). In other words, if $\psi_k(x) \approx 1$ then the dataset $\mathcal{L}_k$ strongly indicates that $x \in \mathcal{R}_T(\Omega)$. Likewise, if $\psi_k(x) \approx 0$ then the dataset $\mathcal{L}_k$ strongly indicates that $x \notin \mathcal{R}_T(\Omega)$.

The informativeness of the samples in $\mathcal{S}$ are quantified using the Shannon entropy

$$J_E(S) = -\sum_{x \in S} \sum_{y \in \{-1,+1\}} p_k(y|x) \log_2 |p_k(y|x)| \tag{7}$$

where $p_k(+1|x) = \frac{1}{2} + \frac{1}{2}\psi_k(x)$ is the probability that $x \in \mathcal{R}_T(\Omega)$ based on the belief ψ_k, and $p_k(-1|x) = \frac{1}{2} - \frac{1}{2}\psi_k(x)$ is the probability that $x \notin \mathcal{R}_T(\Omega)$ based on ψ_k. The entropy (7) is large for samples $x \in \mathcal{S}$ with high uncertainty, $\psi_k(x) \approx 0.5$.

The samples $x \in \mathcal{S}$ can have redundant information, which is quantified by the mutual information function $J_D(\mathcal{S})$

defined by

$$J_D(\mathcal{S}) = \frac{1}{|\mathcal{L}_k|} \sum_{i=1}^{|\mathcal{L}_k|} \max_j D_{ij}^k(\mathcal{S}), \quad (8)$$

where $D^k(\mathcal{S}) \in \mathbb{R}^{|\mathcal{L}_k| \times |\mathcal{S}|}$ is a non-negative matrix whose element $D_{ij}^k(\mathcal{S}) \geq 0$ contains the amount of mutual information between the ith labeled sample from $\mathcal{L}_k$ and the jth unlabeled one from $\mathcal{S}$. This mutual information can be estimated by computing the relative distance between the distributions of the ith and jth samples conditional upon the classifier ψ_k. For example, one can employ the Kullback-Liebler (KL) divergence metric

$$D_{ij}^k = -\sum_{y \in \{-1,+1\}} p_y^i \log_2 \left| \frac{p_y^j}{p_y^i} \right|,$$

where $p_y^i = p_k(y|x_i)$. Using the entropy (7) and mutual information (8), we pose the informativeness-redundancy trade-off as a cardinality constrained submodular maximization problem.

The B most useful samples in $\mathcal{S}$ are obtained by solving:

$$\mathcal{L}_k' = \underset{\mathcal{S}_0 \subseteq \mathcal{S}: |\mathcal{S}_0| \leq B}{\arg\max} \; J(\mathcal{S}_0). \quad (9)$$

where the objective function is given by

$$J(\mathcal{S}_0) \triangleq J_E(\mathcal{S}_0) + J_D(\mathcal{S}_0). \quad (10)$$

The component J_E encourages samples with new information to be chosen, while J_D penalizes redundancy compared with prior labeled samples. We have noted that, empirically, adding J_D does not always have a significant effect on the quality of learning, and can be considered optional.

Unfortunately, problem (9) is NP-hard and thus no polynomial-time algorithm exists with an approximation factor better than $(1 - 1/e)$, where $\ln(e) = 1$; see [39]. However, if J_E and J_D belong to the class of submodular functions, one can provide polynomial-time algorithms that generate near-optimal solutions. We take the following definition from [25].

Definition 2. *Consider a set function $J : \mathbf{2}^{\mathcal{W}} \to \mathbb{R}$ that maps subsets of a finite set $\mathcal{W}$ to the reals. The function J is:*

(P1) ***normalized****: if $J(\emptyset) = 0$.*

(P2) ***monotone****: if $J(\mathcal{W}_1) \leq J(\mathcal{W}_2)$ for any $\mathcal{W}_1 \subseteq \mathcal{W}_2 \subseteq \mathcal{W}$.*

(P3) ***submodular****: if, for every $\mathcal{W}_1 \subseteq \mathcal{W}_2 \subseteq \mathcal{W}$, and $w \in \mathcal{W} \setminus \mathcal{W}_2$, the inequality*

$$J(\mathcal{W}_1 \cup \{w\}) - J(\mathcal{W}_1) \geq J(\mathcal{W}_2 \cup \{w\}) - J(\mathcal{W}_2)$$

is satisfied.

The problem (9) can be solved near-optimally using a greedy approach that iteratively extracts the sample $x \in \mathcal{S}$ which increases J the most, until B such samples are selected; see Algorithm 1. We refer to

$$\Delta(x|\mathcal{S}; J) \triangleq J(\mathcal{S} \cup \{x\}) - J(\mathcal{S}) \quad (11)$$

as the marginal gain of a function J for the sample x with the prior set $\mathcal{S}$.

Algorithm 1 Greedy

```
Input: Set functions $J_E$, $J_D$
Input: Unlabeled set, $\mathcal{S}$
Input: Batch size, $B$
Output: Actively learned samples, $\mathcal{L}'_{B,\mathsf{G}}$
1: $\mathcal{L}'_{k,\mathsf{G}} \leftarrow \emptyset$
2: for $k = 1 : B$ do
3:     $x^\star \leftarrow \arg\max_{x \in \mathcal{S} \setminus \mathcal{L}'_{k,\mathsf{G}}} \Delta(x|\mathcal{L}'_{k,\mathsf{G}}; J)$
4:     $\mathcal{L}'_{k,\mathsf{G}} \leftarrow \mathcal{L}'_{k,\mathsf{G}} \cup \{x^\star\}$
5: end for
6: return $\mathcal{L}'_{B,\mathsf{G}}$
```

The following theorem quantifies the relative suboptimality of a solution generated by Algorithm 1.

Theorem 1. *Let B be the batch size, and J_E and J_D be defined as in* (7) *and* (8)*, respectively. The relative suboptimality for any solution $\mathcal{L}'_{B,\mathsf{G}}$ obtained by solving* (9) *using Algorithm 1 is*

$$\frac{J^\star - J(\mathcal{L}'_{B,\mathsf{G}})}{J^\star} \leq \frac{1}{e}, \quad (12)$$

where $J^\star$ denotes the optimal cost. The time complexity incurred is $\mathbf{O}(|\mathcal{S}|B^2)$.

Proof: The proof involves demonstrating that the choice of $J(\cdot)$ in (10) exhibits the properties (P1)–(P3) in Definition 2. Selecting no sample from $\mathcal{S}$ implies that $J_E(\emptyset) = 0$ and $J_D(\emptyset) = 0$; thus, $J(\cdot)$ is normalized and (P1) is satisfied. We know from [40] that the entropy function (7) (for finite, discrete-valued random vectors) and the facility location function that has the form (8) (with non-negative D_{ij}^k since KL divergence is non-negative) are monotone submodular. Since the sum of monotone functions is monotone, and the sum of submodular functions is submodular, $J(\cdot)$ is monotone submodular; property (P2) and (P3) are satisfied. The inequality (12) follows immediately from [25] as a consequence of (P1)–(P3) being satisfied. The time complexity is derived in [28, Remark 4].

■

Remark 1. One can use (P1)–(P3) to guide the selection of J_E and J_D. Any J_E and J_D that represent information content and sample divergence that are normalized, monotone, and submodular can be used in lieu of (7) and (8) with identical solution guarantees. Our particular selection of J_E and J_D involves simple computations and can be seamlessly integrated with most classification frameworks, as long as they generate probabilistic predictions or can be modified to do so.

The above steps continue until a termination criterion such as the total number of iterations or the size of the final labeled set is achieved. With the final labeled set, one can generate inner approximations of the reachable set using asymmetric costs or by selecting sub-level sets of the decision function that guarantee no infeasible training or validation sample is labeled feasible [17].

The natural way to represent a set using a function is through indicator functions. However, indicator functions are not continuous, which is a challenge for classifiers that generate continuous decision functions. Thus, the following theorem provides conditions under which the reachable set is a sub-level set of a continuous function.

Theorem 2. *Let the dynamics f be continuous, and the sets $\mathbb{X}$, $\mathbb{U}$, and Ω be compact. There exists a continuous function ζ such that boundary of the reachable set is its zero level set; that is, $\partial\mathcal{R}_T(\Omega) = \{x : \zeta(x) = 0\}$.*

Proof: We prove the theorem for two scenarios: (i) when the system is autonomous; that is, $f(x,u) = f(x,0)$ since $u = 0$, and (ii) when the system is non-autonomous.

Since Ω is compact, there exists a continuous function ζ_0 such that $\Omega = \{x : \zeta_0(x) \le 0\}$ is the sub-zero level-set of ζ_0. An example of such a ζ_0 is $\varkappa_\Omega(x) - 1$, where $\varkappa_\Omega$ is the Minkowski functional of Ω [41].

For autonomous systems, we define

$$\zeta_k(x) = \max\{\varkappa_{\mathbb{X}}(x) - 1, \zeta_{k-1}(f(x))\}, \tag{13}$$

for $k = 1, \cdots, T$ where $\varkappa_{\mathbb{X}}$ is the Minkowski function of the state constraints $\mathbb{X}$. Consider $k = 1$. By construction, $\{x : \zeta_0(x) \le 0\}$ defines the target set Ω and

$$\mathbb{X} = \{x : \varkappa_{\mathbb{X}}(x) \le 1\}.$$

Thus, $x \in \Omega \cap \mathbb{X}$ if and only if $\zeta_1(x) \le 0$, where ζ_1 is defined by (13). Since the maximum of two continuous functions is continuous, $\zeta_1(x)$ is continuous. Using induction for $k = 1, \ldots, T$, we deduce that ζ_T is continuous and satisfies $\zeta_T(x) \le 0$ if and only if $x \in \mathcal{R}_T(\Omega)$.

For non-autonomous systems, we define

$$\begin{aligned}\zeta_k(x) &= \max\{\varkappa_{\mathbb{X}}(x) - 1, \min_{u\in\mathbb{U}} \zeta_{k-1}(f(x,u))\}, \\ &= \max\{\varkappa_{\mathbb{X}}(x) - 1, V^\star(x)\},\end{aligned} \tag{14}$$

for $k = 1, \cdots, T$ where the optimal value-function $V^\star(x)$ of the following parametric program

$$V^\star(x) = \min_{u\in\mathbb{U}} \zeta_{k-1}(f(x,u)).$$

Next, we show that $V^\star(x)$ is continuous (even though the optimal controller $u^\star(x)$ is not necessarily continuous). Consider the constant set-valued function $U : \mathbb{X} \to \mathbb{U}$ given by $U(x) = \mathbb{U}$ for all $x \in \mathbb{X}$. Clearly, the function $U(x)$ is locally compact at every $x \in \mathbb{X}$ since $\mathbb{U}$ is a compact set. Thus, $V^\star(x)$ is lower semi-continuous [42, Lemma 5.3b]. Likewise, the constant function U is inner semi-continuous. Thus, $V^\star(x)$ is upper semi-continuous [42, Lemma 5.4a]. Since $V^\star(x)$ is both upper and lower semi-continuous, it is continuous. Using identical inductive arguments as the autonomous case, we deduce that the level-set function ζ_T is continuous and satisfies $\zeta_T(x) \le 0$ if and only if $x \in \mathcal{R}_T(\Omega)$. ■

The above theorem enables the following guarantee on the learned reachable set: namely, that a good estimate of the reachable set is possible by using learners that can approximate continuous functions arbitrarily well.

Corollary 1. *Suppose the conditions of Theorem 2 hold. Let $\mathcal{H}$ be a hypothesis class that induces decision functions which are dense in the space of continuous functions on the compact metric space $(\mathbb{X}, d)$. Let $\hat\zeta \subset \{\zeta > 0\}$ and $\check\zeta \subset \{\zeta < 0\}$ denote compact subsets of super and sub-level sets of the reachable set boundary. Then any learner $\psi \in \mathcal{H}$ separates $\hat\zeta$ and $\check\zeta$.*

Proof. This is a direct consequence of the sets $\hat\zeta$ and $\check\zeta$ being disjoint. Thus, $d(\hat\zeta, \check\zeta) > 0$. Therefore, $\mathbf{1}_{\hat\zeta} - \mathbf{1}_{\check\zeta}$ is a continuous function on $\hat\zeta \cup \check\zeta$ which can be extended to $\mathbb{X}$ using Whitney's extension theorem [17, Theorem 1]. Consequently, the zero level set of $\mathbf{1}_{\hat\zeta} - \mathbf{1}_{\check\zeta}$ is the true separating boundary, which can be generated by a learning ψ as it belongs to the hypothesis class $\mathcal{H}$ defined in the theorem statement. This concludes the proof. □

Remark 2. The hypothesis class $\mathcal{H}$ in Corollary 1 includes learners with universal kernels [43], or universal neural approximators [44].

Remark 3. The sampling method provided in this paper facilitates learning without extensive sampling; providing guarantees on the learning quality is beyond the scope of this paper as it depends, among other things, on the sampling method employed to generate $\mathcal{S}$, the hypothesis class chosen by the user, the complexity (e.g. VC dimension) of the problem class.

Remark 4. The decision function of many classification algorithms can be written using closed-form analytical expressions. For example, an SVM bi-classifier would have a decision boundary given by $\sum y_i \alpha_i \mathcal{K}(x_i, x) = 0$, where y is the label vector, α is the vector of Lagrange multipliers, x_i is the i-th feature, and $\mathcal{K}$ is the kernel matrix; for more details, see [17].

IV. Efficient Implementations for Large B or $\mathcal{S}$

One generally wants the cardinality of $\mathcal{S}$ to be large to ensure that the unlabeled samples are well dispersed throughout $\mathbb{X}$. Since Algorithm 1 exhibits a complexity $\mathbf{O}(|\mathcal{S}|B^2)$, increasing the cardinality of $\mathcal{S}$ and B could increase solver execution time exorbitantly. Variants of Algorithm 1 that get around this issues without major losses in performance can be obtained by exploiting random subsampling and distributed computations. We present two such variants: (i) a stochastic variant of Algorithm 1 to facilitate active learning on large sets of unlabeled samples with time complexity linear in $|\mathcal{S}|$ and B; and, (ii) for cases when $N_c \ge 2$ clusters are available for distributed implementation, we provide a distributed algorithm that exhibits a time complexity linear in $|\mathcal{S}|/N_c$.

A. Stochastic Greedy Algorithm

The stochastic greedy variant of Algorithm 1 is presented herein. The method involves curtailing the number of marginal gain computations. Instead of computing the marginal gain of every sample in $\mathcal{S}$, the stochastic greedy method selects a random subset $\mathcal{S}_q \subset \mathcal{S}$ containing q distinct elements of $\mathcal{S}$. The random subset $\mathcal{S}_q$ is resampled at each iteration. The maximizer of the marginal gain using only elements from $\mathcal{S}_q$ is chosen for appending to the set of greedy solutions. As argued in [30], the method is effective because, for a given $\varepsilon > 0$, by choosing q (function of ε) large enough, one can show that the random subset $\mathcal{S}_q$ will contain elements contained in the

Algorithm 2 Stochastic Greedy

```
Input: Set functions $J_E$, $J_D$
Input: Unlabeled set, $\mathcal{S}$
Input: Batch size, $B$
Input: Approximation tolerance, $\varepsilon$
Output: Actively learned samples, $\mathcal{L}'_{B,\mathsf{SG}}$
1: $q \leftarrow (|\mathcal{S}|/B)\ln(1/\varepsilon)$
2: $\mathcal{L}'_{k,\mathsf{SG}} \leftarrow \emptyset$
3: for $k = 1 : B$ do
4:     $S_q \leftarrow$ random subset of $|\mathcal{S}| \setminus \mathcal{L}'_{k,\mathsf{SG}}$ with $q$ elements
5:     $x^\star \leftarrow \arg\max_{x\in S_q} \Delta(s|\mathcal{L}'_{k,\mathsf{SG}}; J)$
6:     $\mathcal{L}'_{k,\mathsf{SG}} \leftarrow \mathcal{L}'_{k,\mathsf{SG}} \cup \{x^\star\}$
7: end for
8: return $\mathcal{L}'_{B,\mathsf{SG}}$
```

greedy optimal solution with probability $1-\varepsilon$. The pseudocode is provided in Algorithm 2.

The following theorem shows that the stochastic greedy algorithm may result in near-optimal solutions with a time complexity that depends linearly (instead of quadratically) on the batch-size B, by ensuring the random subsets S_q selected in each iteration satisfy a pre-computable cardinality constraint.

Theorem 3. *Let B be the batch size, and the set functions J_E and J_D be defined as in* (7) *and* (8)*, respectively, and $J^\star$ is the optimal solution for* (9)*. Let $0 < \varepsilon \ll 1$ and fix $q = (|\mathcal{S}|/B)\ln|1/\varepsilon|$. Then, any solution obtained by solving Algorithm 2 satisfies*

$$\frac{J^\star - \mathbb{E}\left[J(\mathcal{L}'_{B,\mathsf{SG}})\right]}{J^\star} \le \frac{1}{e} + \varepsilon, \tag{15}$$

with a time complexity of $\mathbf{O}(|\mathcal{S}|B\ln|1/\varepsilon|)$.

Proof: From the proof of Theorem 1, we know that $J(\cdot)$ is normalized, monotone, and submodular. From monotonicity and normalization, we get $J(\emptyset) = 0$ and $J(\emptyset \cup \mathcal{S}_0) \ge J(\emptyset)$ for any $\mathcal{S}_0$. Hence, J is non-negative. Then (15) follows from [30, Theorem 1]. Since the cardinality of $\mathcal{S}_q$ is q, replacing this quantity instead of $|\mathcal{S}|$ in the complexity $\mathbf{O}(|\mathcal{S}|B^2)$ of Algorithm 1 yields the desired complexity. ■

B. Distributed Implementation

For sampling patterns that grow exponentially with dimension (for example, regular grids), a computational expenditure of $\mathbf{O}(|\mathcal{S}|)$ may be impractical. In such scenarios, the use of distributed computing provides a computationally efficient way of implementing greedy algorithms for solving (9) without completely losing optimality guarantees [31], [45]. In particular, we consider the RANDGREEDI algorithm of [45].

Suppose there are N_c nodes/clusters where the distributed algorithm can be deployed. The RANDGREEDI algorithm proceeds as follows. At each node $1 \le k \le N_c$, a subset $\mathcal{S}_k$ is formed by randomly selecting elements from the unlabeled set $\mathcal{S}$ using a uniform distribution. Once $\mathcal{S}$ is randomly partitioned into $\{\mathcal{S}_k\}_{k=1}^{N_c}$, each node runs Algorithm 1 on its corresponding unlabeled set. Once all N_c nodes have computed their greedy solutions, these solutions $\mathcal{L}'_{k,\mathsf{G}}$ are returned to one of the nodes or a centralized processing unit. At this centralized node, two operations are performed: (i) all greedy solutions are composed into a set $\mathcal{S}_\infty$, using which a greedy solution $\mathcal{L}'_{\infty,\mathsf{G}}$ is computed, and (ii) the best among the k distributed solutions, denoted $\mathcal{L}'_{k^*,\mathsf{G}}$ is computed. The better of these two centralized solutions is returned as the final solution $\mathcal{L}'_{B,\mathsf{DG}}$. The pseudocode is presented in Algorithm 3.

Algorithm 3 Distributed Greedy Algorithm (RANDGREEDI)

```
Input: Set functions $J_E$, $J_D$
Input: Unlabeled set, $\mathcal{S}$
Input: Batch size, $B$
Input: Number of clusters, $N_c$
Input: $\mathcal{S}_k = \emptyset$ for $1 \le k \le N_c$
Output: Actively learned samples $\mathcal{L}'_{B,\mathsf{DG}}$
–DISTRIBUTED–
1: for $x \in \mathcal{S}$ do
2:     Select $k \in \{1, 2, \ldots, N_c\}$ uniformly at random
3:     $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{x\}$
4: end for
5: for $k = 1 : N_c$ do
6:     $\mathcal{L}'_{k,\mathsf{G}} \leftarrow$ greedy solution from $\mathcal{S}_k$
7: end for
–CENTRALIZED–
8: $\mathcal{S}_\infty \leftarrow \bigcup_{k=1}^{N_c} \mathcal{L}'_{k,\mathsf{G}}$
9: $\mathcal{L}'_{\infty,\mathsf{G}} \leftarrow$ greedy solution from $\mathcal{S}_\infty$
10: $\mathcal{L}'_{k^*,\mathsf{G}} \leftarrow \arg\max_{1\le k\le N_c} \left\{J\left(\mathcal{L}'_{k,\mathsf{G}}\right)\right\}$
11: $\mathcal{L}'_{B,\mathsf{DG}} \leftarrow \arg\max \left\{J\left(\mathcal{L}'_{\infty,\mathsf{G}}\right), J\left(\mathcal{L}'_{k^*,\mathsf{G}}\right)\right\}$
12: return $\mathcal{L}'_{B,\mathsf{DG}}$
```

The following theorem from [45] demonstrates that randomly partitioning into N_c nodes/clusters and solving the distributed problem results in lowering the optimality guarantee by a (constant) factor of 2.

Theorem 4. *Let B be the batch size, and the set functions J_E and J_D be defined as in* (7) *and* (8)*, respectively, and let $J^\star$ be the optimal solution for* (9)*. Let $N_c \ge 2$ denote the number of distributed nodes, and suppose $|\mathcal{S}| \gg N_c^2 B$. Then, any solution obtained by solving Algorithm 3 satisfies*

$$\frac{J^\star - \mathbb{E}\left[J(\mathcal{L}'_{B,\mathsf{DG}})\right]}{J^\star} \le \frac{1+e}{2e}. \tag{16}$$

with an expected time complexity of $\mathbf{O}(\gamma_1 B^2)$ *for each node, where* $\gamma_1 = \lceil|\mathcal{S}|/N_c\rceil$.

Proof: The optimality guarantee follows from [45, Theorem 5] with simple algebraic manipulations. To derive the time complexity, consider that each distributed node has a ground set of cardinality (in expectation) at most γ_1, which implies that running the algorithm on node k has an expected time complexity of $\mathbf{O}(\gamma_1 B^2)$, from Theorem 1. For the centralized node, along with the distributed greedy operation, it also computes a greedy solution with the union set $\mathcal{S}_\infty$. The largest cardinality $\mathcal{S}_\infty$ can have is $N_c B$, since each greedy solution $\mathcal{L}'_{k,\mathsf{G}}$ has cardinality B, which means the central node needs to obtain (in the worst-case) a greedy solution with a ground set of size N_cB, which incurs a time complexity of $\mathbf{O}(N_cB^3)$. Since $N_cB \ll |\mathcal{S}|/N_c$, this time is dwarfed by the greedy search and so the overall complexity is still $\mathbf{O}(\gamma_1 B^2)$. ■

As evident from Theorem 4, the condition $|\mathcal{S}| \gg N_c^2 B$ implies that the unlabeled set is massive, if the number of clusters is large. Without large N_c, it appears that using Algorithm 2 is more efficient and provides better optimality guarantees than Algorithm 3.

V. Learning with Strong and Weak Oracles

In the previous sections, we assumed that the oracle is infallible; that is, $\mathcal{O}(x)$ is correct for all $x \in \mathbb{X}$. In this section, we consider the scenario when we have access to two oracles: a *strong* oracle $\mathcal{O}_S$ that incurs a higher cost to evaluate but is infallible, and a *weak* oracle $\mathcal{O}_W$ whose evaluation complexity is cheap, but which does not always provide the correct label (that is, it is fallible). We propose an active learning algorithm that leverages the weak oracle $\mathcal{O}_W$ to reduce the total labeling expenditure. We do so by lowering the number of queries to the expensive strong oracle $\mathcal{O}_S$. To this end, we adopt the disagreement-based active learning (DBAL) philosophy described in [34].

The concept of a weak (or approximate) oracle is defined below.

Definition 3 (Weak Oracle)**.** *A weak oracle $\mathcal{O}_W(x)$ is an indicator function for a set $\hat{\mathcal{R}}_T(\Omega)$ that approximates the actual reachable set $\mathcal{R}_T(\Omega)$*

$$\mathcal{O}_W(x) = \begin{cases} +1 \text{ if } x \in \hat{\mathcal{R}}_T(\Omega) \\ -1 \text{ if } x \notin \hat{\mathcal{R}}_T(\Omega). \end{cases} \tag{17}$$

To illustrate the concept of weak and strong oracles, consider the following example.

Example 5. Recall the oracle defined in Example 1 which evaluates the indicator function of the reachable set (2) by solving a finite-time optimal control problem (5). An example of a weak oracle is using an iterative method to solve the optimization problem (5) where the terminal condition has a large tolerance. Whereas, a strong oracle could be obtained by solving the optimization problem (5) using an active-set solver which produces the exact solution after a small number of iterations but requires more compute and cannot be deployed on low-end processors. □

We define the *disagreement region* $\mathbb{X}_D \subset \mathbb{X}$ as the set of states where the strong $\mathcal{O}_S$ and weak $\mathcal{O}_W$ oracles disagree

$$\mathbb{X}_D = \mathcal{R}_T(\Omega) \triangle \hat{\mathcal{R}}_T(\Omega). \tag{18}$$

We prove the following property about $\mathbb{X}_D$.

Corollary 2. *If $\hat{\mathcal{R}}_T(\Omega)$ is compact, the set $\mathbb{X}_D$ has an interior and exterior.*

Proof: Using Theorem 2, we can construct continuous functions ζ and $\hat{\zeta}$ such that $\partial\mathcal{R}_T(\Omega) = \{x : \zeta(x) = 0\}$ and $\partial\hat{\mathcal{R}}_T(\Omega) = \{x : \hat{\zeta}(x) = 0\}$. Then, we can write

$$\begin{aligned} \mathbb{X}_D &= \mathcal{R}_T(\Omega) \triangle \hat{\mathcal{R}}_T(\Omega) \\ &= (\hat{\mathcal{R}}_T(\Omega) \setminus \mathcal{R}_T(\Omega)) \cup (\mathcal{R}_T(\Omega) \setminus \hat{\mathcal{R}}_T(\Omega)). \end{aligned}$$

This can be written as the zero level set of the function

$$\zeta_D := \min\Big\{\max\{\hat{\zeta}, -\zeta\}, \max\{\zeta, -\hat{\zeta}\}\Big\}$$

which is continuous. Therefore, sub- and super- level sets of ζ_D are interior and exterior regions of $\mathbb{X}_D$. ■

By Corollary 2, one can generate inner $\mathcal{D}^- \subseteq \mathbb{X}_D$ and outer $\mathcal{D}^+ \supseteq \mathbb{X}_D$ approximations of the disagreement region $\mathbb{X}_D$. Subsequently, we can use the weak oracle (17) to define a strong oracle, as shown in the following theorem.

Theorem 5. *The oracle*

$$\mathcal{O}_{S,W} = \begin{cases} \mathcal{O}_W(x), & \text{for every } x \in \mathbb{X} \setminus \mathcal{D}^+ \\ -\mathcal{O}_W(x), & \text{for every } x \in \mathcal{D}^- \\ \mathcal{O}_S(x), & \text{otherwise,} \end{cases} \tag{19}$$

is infallible. Furthermore, for x uniformly sampled on $\mathbb{X}$, the expected number of queries to $\mathcal{O}_S$ is $\mu_L(\mathcal{D}^+ \setminus \mathcal{D}^-)/\mu_L(\mathbb{X})$, where μ_L is the Lebesgue measure.

Proof: For any $x \in \mathcal{D}^-$, the weak oracle is reliably incorrect, that is, for these x, we know that $\mathcal{O}_W \neq \mathcal{O}_S$. Since the underlying classification problem, to be solved using the oracles, is binary, $\mathcal{O}_W = -\mathcal{O}_S$. For any $x \in \mathbb{X} \setminus \mathcal{D}^+$, the x are outside the disagreement region, so $\mathcal{O}_W$ can be trusted. Hence, for these x, we know that $\mathcal{O}_W = \mathcal{O}_S$. Clearly, for $x \in \mathcal{D}^+ \setminus \mathcal{D}^-$, the weak oracle may or may not be correct, so we must resort to $\mathcal{O}_S$ for true labels. Since this region has volume $\mu_L(\mathcal{D}^+ \setminus \mathcal{D}^-)$, the probability of sampling from a uniform distribution within this region in an ambient space $\mathbb{X}$ of volume $\mu_L(\mathbb{X})$ is given by the ratio of the two volumes. This concludes the proof. ■

To leverage the insights in the above result, our proposed DBAL framework involves constructing two learners. One of these learners is trained to identify inner $\mathcal{D}^-$ and outer $\mathcal{D}^+$ approximations of the disagreement region $\mathbb{X}_D$; this learner is therefore referred to as the *disagreement learner*. The *primary learner* (the same learner that is studied in Section III) leverages the disagreement learner's knowledge of where each oracle needs to be queried in order to actively learn the safe set. We modify the algorithm proposed in [34] in order to solve this disagreement-based active learning problem.

Concretely, we perform two rounds of active learning. In the first round, we seek to identify active samples that are most useful for estimating the boundary of $\mathbb{X}_D$ via the disagreement learner $\psi_D : \mathbb{X} \to [0, 1]$. The classifier ψ_D can be interpreted the belief $\psi_D(x) \in [0, 1]$ that a sample $x \in \mathbb{X}$ is in the disagreement region $\mathbb{X}_D$. Therefore, we can use an oracle of the form $\mathcal{O}_{S,W}(x)$. By using cost-sensitive learning or level sets of ψ_D, we construct two approximators $\hat{\mathcal{D}}^+ = \{x : \psi_D(x) > T^+\}$ and $\hat{\mathcal{D}}^- = \{x : \psi_D(x) < T^-\}$, that are inner and outer approximations of $\mathbb{X}_D$. For example, the threshold T^- can be selected so that $\hat{\mathcal{D}}^-$ has no false negatives i.e. $\mathcal{O}_W(x) \neq \mathcal{O}_S(x)$ for all $x \in \hat{\mathcal{D}}^- \cap \mathcal{L}_k$. Similarly, the threshold T^- could be selected so that $\hat{\mathcal{D}}^+$

does not contain any false positives i.e. $\mathcal{O}_W(x) = \mathcal{O}_S(x)$ for all $x \in \hat{\mathcal{D}}^+ \cap \mathcal{L}_k$. Once these sets are identified, the same algorithm as in Section III can be employed to identify the safe set, with the number of strong oracle calls reduced using the new oracle (19) with $\mathcal{D}^+$ and $\mathcal{D}^-$ replaced by their estimates $\hat{\mathcal{D}}^+$ and $\hat{\mathcal{D}}^-$, respectively. The pseudocode of the proposed algorithm is provided in Algorithm 4.

Remark 5. We assume that estimating the disagreement region requires fewer labels than identifying the reachable set and that the cost of always calling the strong oracle is exorbitant. Otherwise, one could use active learning using only the strong oracle.

Remark 6. The theoretical guarantees of suboptimality for the disagreement learner and the primary learner remain the same as in Theorem 1.

Algorithm 4 Proposed DBAL Algorithm

```
Input: Set of unlabeled samples, $\mathcal{S}$
Input: Initial labeled samples for disagreement, $\mathcal{L}_{0,D}$
Input: Number of strong oracle calls, $N_{s,D}$
Input: Batch sizes, $B_D$
Input: Oracles: $\mathcal{O}_W$, $\mathcal{O}_S$
Input: Number of active samples to select, $N_s$
Input: Batch sizes, $B$
Input: Oracles: $\mathcal{O}_{S,W}$ in (19) using $\hat{\mathcal{D}}^+$ and $\hat{\mathcal{D}}^-$
Output: Disagreement learner $\psi_D$, primary learner $\psi$
–DISAGREEMENT LEARNER–
1: $\psi_D \leftarrow$ disagreement learner using oracle $\mathcal{O}_S/\mathcal{O}_W$, batch
   size $B_D$, final number of labeled samples $N_{s,D}$
–PRIMARY LEARNER–
2: $\hat{\mathcal{D}}^+ \leftarrow$ outer approximation of $\mathbb{X}_D$ from $\psi_D$
3: $\hat{\mathcal{D}}^- \leftarrow$ inner approximation of $\mathbb{X}_D$ from $\psi_D$
4: $\mathcal{L}_0 \leftarrow$ labeled samples obtained after disagreement learn-
   ing phase
5: $\psi \leftarrow$ primary learner using oracle $O_{S,W}$, batch size $B$,
   final number of labeled samples $N_s$
6: return $\psi_D$, $\psi$
```

VI. Simulation Results

A. Highly imbalanced dataset

We illustrate our proposed approach on the system:

$$x_{1,t+1} = x_{1,t} + 0.05\left(-x_{2,t} + 0.5(1 + x_{1,t})u_t\right)$$
$$x_{2,t+1} = x_{2,t} + 0.05\left(x_{1,t} + 0.5(1 - 4x_{2,t})u_t\right),$$

with $\mathbb{U} = \{u \in \mathbb{R} : |u| \le 2\}$ [46]. We know that $\mathbb{X} = \{x \in \mathbb{R}^2 : \|x\|_\infty \le 4\}$. The oracle in this example is a nonlinear MPC that returns $+1$ when a feasible solution is found to the problem (5) from a given initial condition in $\mathbb{X}$, and -1 otherwise. A complete description of the implementation is provided in [28].

The unlabeled set of samples is generated using low-discrepancy Halton samples [17]; $|\mathcal{S}| = 5 \times 10^3$. We fix the batch size to be $B = 100$ samples. The total number of active samples to be selected is set to $N_s = 1000$. The classifier ψ_k is selected to be a support vector machine (SVM) with Gaussian radial basis function kernels and Platt scaling to generate probabilistic outputs. The learner is implemented in MATLAB R2019a via the Statistics and Machine Learning toolbox with automatic hyperparameter tuning via Bayesian optimization.

The set $\mathcal{L}_0$ with 105 samples (of which 8 are feasible) is illustrated in Fig. 1[A]. This figure also shows the active samples (red squares) selected over all ten iterations. As expected, the AL procedure selects states that are the most uncertain, while ignoring samples far from the classifiers' decision boundaries as they are most likely to be infeasible. The potential of our proposed method is illustrated in Fig. 1[B], where we observe that active learning outperforms passive learning by lowering the number of samples by an order of magnitude and despite that, producing a much better representation of the true reachable set. It is also clear from this figure that relying on Lipschitz constants and solving for domains of attraction with classical control-theoretic analytical tools [47] could result in a very conservative set estimate.

Although imbalance correction of the data is not explicitly required, since AL intrinsically selects samples near higher uncertainty regions, the imbalance improves as the likelihood of labeling infeasible and feasible samples are about the same in those regions. We observe a clear improvement of balance in the data in Fig. 1[C]; from a meagre 7% of feasible samples, we improve the balance of the data close to 50% after ten iterations of AL. Of course, the limitation of AL methods is the training time: Fig. 1[D] demonstrates this trend in the cumulative training times. The resultant cumulative training time of 6 s in AL is almost three times more than the classifier trained passively (2.34 s).

Another notable benefit of our proposed approach is its generalizability. We test the active learning algorithm using random forests, SVMs, and shallow neural networks. We see in Fig. 2 that the precision-recall (PR) curves for all three classifiers with 1000 active samples (continuous lines) exhibit significantly higher AUCs (area under the curves) in comparison with their passive counterparts (dotted lines) in spite of 10× more samples for training.

We also compare the performance of different AL algorithms: marginal SVM [48], uncertainty sampling, batch-mode active learning using BatchRank and BatchRand algorithms [23], and the proposed method. Points of comparison and the performances of these three classifiers are reported in Table I. Myopic algorithms (that believe only points near the decision boundary are informative) such as marginal SVM and uncertainty sampling are outperformed by algorithms that explicitly minimize redundancy. Thus, non-myopic algorithms such as BatchRank, BatchRand, and our proposed approach, produce large areas under their precision-recall curves. The precision-recall characteristics are computed based on a uniform 300×300 grid within $[-0.75, 0.75]^2$ to test classification performance near the true decision boundary. Both the BatchRank and BatchRand algorithms rely on convex relaxations of an NP-hard integer quadratic programming problem. For example, the BatchRand algorithm exploits a semidefinite programming (SDP) relaxation: the solution of

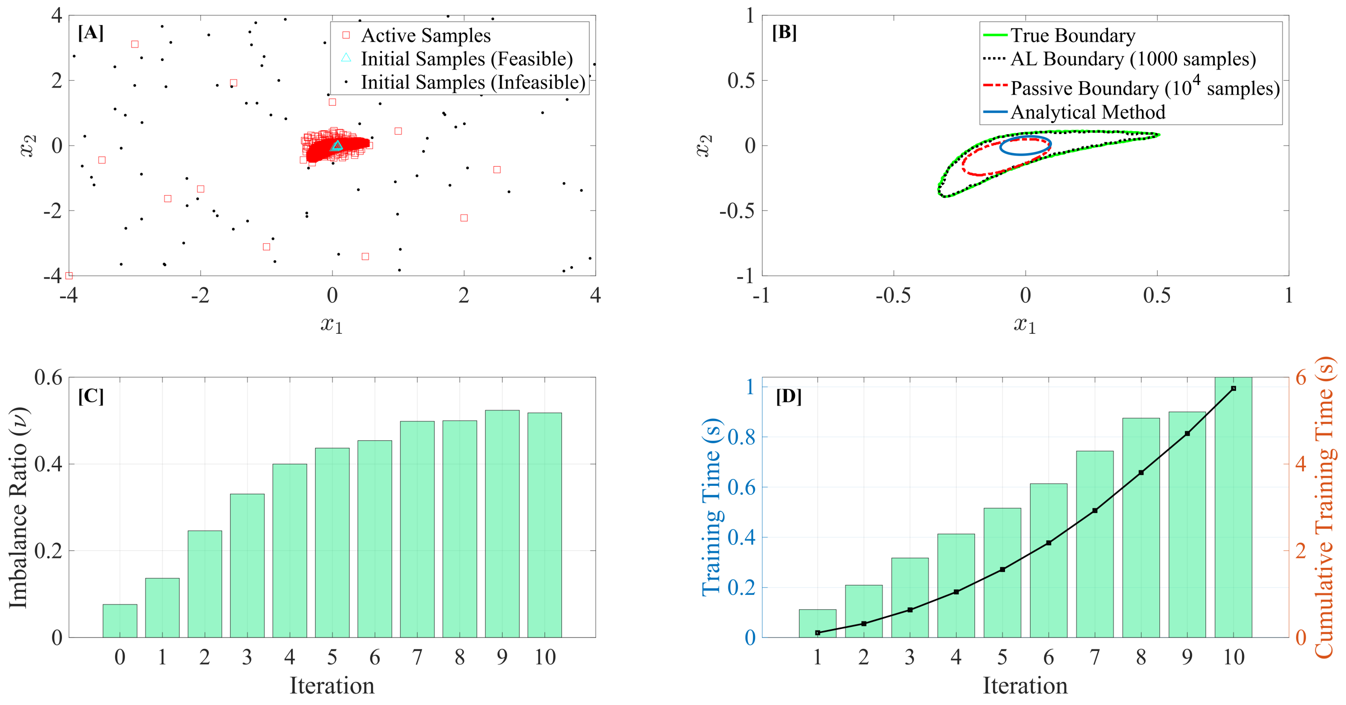

Fig. 1. [A] Initial random samples (infeasible: dots; feasible: triangles). [B] Set estimation (green continuous line) using active learning (black dotted line), passive learning (red dashed line), and analytical control-theoretic methods (blue continuous ellipse). [C] Imbalance ratio (ν) over iterations. [D] Training time and cumulative training time in seconds.

TABLE I
COMPARISON OF AL ALGORITHMS

Algorithm	AUC	CPU Time [s]	Optimization Method	Solution Guarantees	Complexity
Marginal SVM	0.772	<1	n/a	n/a	$\mathbf{O}(\lvert\mathcal{S}\rvert \ln \lvert\mathcal{S}\rvert)$
Uncertainty Sampling	0.924	<1	n/a	n/a	$\mathbf{O}(\lvert\mathcal{S}\rvert \ln \lvert\mathcal{S}\rvert)$
BatchRank	0.955	3.85	integer QP (LP relaxation)	not on true objective function	$\mathbf{O}(\lvert\mathcal{S}\rvert^2)$
BatchRand	0.953	>200	integer QP (SDP relaxation)	not on true objective function	$\mathbf{O}(\lvert\mathcal{S}\rvert^3)$
Proposed	0.957	5.91	submodular max. (greedy)	near-optimal	$\mathbf{O}(\lvert\mathcal{S}\rvert B^2)$

the SDP culminates in a spike in training times from 6 s for our method to over 200 s. BatchRank entails a linear programming relaxation whose solution is equivalent to a greedy selection algorithm; however, since the problem solved in BatchRank is a supermodular maximization, there are no guarantees on BatchRank's solution quality, which we do via Theorem 1. Comparing computational complexities of the algorithms, our proposed approach significantly outperforms the other non-myopic frameworks because $|\mathcal{S}| \gg B$.

Remark 7. We ran an experiment with multiple batch sizes ($B \in \{10, 20, 50, 100, 250, 500\}$), keeping the total number of active samples fixed, and noted that the AUC range on the respective PR curves are within $[0.96, 0.98]$ with the 0.96 result being exhibited by the largest batch size of 500. This is expected, because there are only two active learning iterations in this case, so the effectiveness of acquiring better information is limited.

Remark 8. While it would be interesting to compare the samples chosen by the active learning algorithm to the optimal solution, computing an optimal solution would be possible only for very small $\mathcal{S}$ and B since the underlying sample selection problem is NP-hard.

B. Domain of attraction estimation of neural controllers

In this example, we determine the domain of attraction of reinforcement learning control policies represented by neural approximators. We consider policy iteration-based controllers for the following nonlinear system studied in [49]:

$$\begin{aligned}\dot{x}_1 &= x_1 + x_2 - x_1(x_1^2 + x_2^2)\\ \dot{x}_2 &= -x_1 + x_2 - x_2(x_1^2 + x_2^2) + u.\end{aligned}$$

Approximate dynamic programming is used to construct a neural-network based control policy that stabilizes this system, but a domain of attraction is not specified and the analytical form of the controller (that is, the architecture and weights of the neural net) is not available to us. The state-space under consideration is given by $\mathbb{X} = [-1, 1]^2$ and the control policy is restricted to $|u| \leq 0.5$, although the control bound is unknown to us. We assume that the closed-loop system with the neural control policy is available for simulation.

We use 40 initial samples with labels obtained during the ADP computations. A batch size of 20 is selected, and

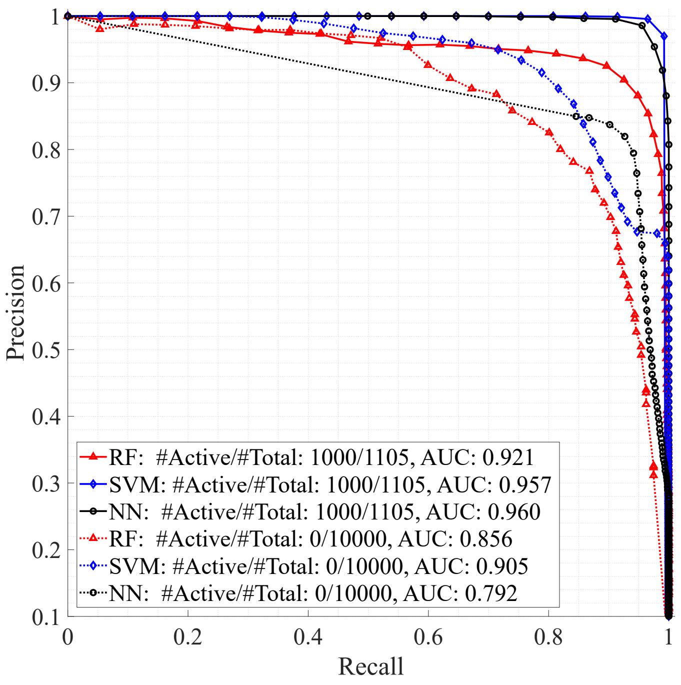

Fig. 2. Comparison of precision-recall (PR) curves for different classifiers: random forests, support vector machines, and neural networks. The number of active samples chosen in each case is 1000, with an initial set of 105 randomly selected samples (PR curve: continuous lines), whereas passive learning is performed with 10^4 samples (PR curve: dotted lines). AUC stands for area under the PR curve.

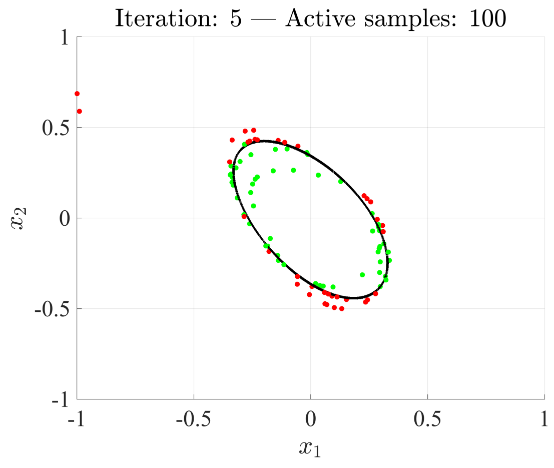

Fig. 3. Domain of attraction of a system driven by a reinforcement learning-based control policy.

$N_s = 100$ active samples are to be computed. The same classifier as in the previous example is used, with a polynomial kernel instead of a radial basis function kernel (determined via hyperparameter optimization). The result of the simulation is illustrated in Fig. 3. The red dots in the figure are initial conditions from which the constrained reinforcement learner could not stabilize the system, and the green dots are initial conditions that are successfully stabilized. The estimate of the reachable set is given by the ellipsoid induced by the low degree polynomial kernel. Cost sensitive learning is used to construct an inner approximation of the domain of attraction (reachable set) of the constrained reinforcement learning-based controller. Randomly selecting 1000 initial conditions within the estimated region and forward simulating up to 200 s verifies the correctness of the estimated domain of attraction (these simulations are not shown).

C. Unknown non-polynomial system with large unlabeled set

We utilize this example to demonstrate the effectiveness of the stochastic and distributed variants of the greedy submodular maximization for large $\mathcal{S}$. We test these methods on a non-polynomial system studied in [13] for which standard sum-of-squares optimization algorithms cannot be used to derive reachable sets. The dynamics of the system under consideration are given by

$$\begin{aligned}
\dot{x}_1 &= x_2 + x_3^2 \\
\dot{x}_2 &= x_3 - x_1^2 - x_1 \sin(x_1) + u_1 \\
\dot{x}_3 &= -x_1 - 2x_2 - x_3 + x_2^3 + \ln\left|\left(\frac{1+x_3}{1-x_3}\right)^{0.1}\right| + u_2
\end{aligned}$$

with stabilizing controls

$$\begin{aligned}
u_1 &= -0.34y_1 - 0.76y_2 + 0.99y_1^2 + 0.16y_1y_2 - 0.80y_2^2 \\
u_2 &= 0.05y_1 + 0.81y_2 - 0.48y_1^2 + 0.95y_1y_2 - 0.92y_2^2
\end{aligned}$$

where $y_1 = x_1 - x_2$, and $y_2 = x_2 - x_3$. The state constraint set is $\mathbb{X} = \{x \in \mathbb{R}^3 : \|x\|_\infty \le 1\}$. We do not assume any knowledge of this system structure and our labeling scheme involves an oracle of the form (6) for the T-step reachable set $\mathcal{R}_T(\Omega)$, where $T = 100$ s and $\Omega = \{x \in \mathbb{X} : \|x\|_2 \le 0.001\}$.

We employ the proposed algorithm with a probabilistic SVM with Platt scaling as in Example 1 with 500 initial labeled samples, and choose 2000 active samples with $B = 100$ samples per batch from a set of $|\mathcal{S}| = 2 \times 10^4$ unlabeled samples. The final inner approximation is constructed with a three-layer neural network with two hidden layers, and a sub-level set is chosen (by altering the bias weight of the output activation layer) that classifies no unreachable point as reachable. The training time of the neural network was 9.39 s using `scikit-learn` in Python 2.7 using limited memory BFGS (L-BFGS) and rectified linear (ReLU) units. The result of AL is illustrated in Fig. 4[A], where the 'true' reachable set generated by simulating over the 10^5 samples drawn from a Halton sequence. The true reachable set is depicted with the gray outer shading and the inner approximation is shown using the red inner volume. The convex hull of the true reachable set and the inner approximation have volumes 1.55 and 1.38 cubic units, respectively, estimated by MATLAB's `boundary` function. As expected, the class-imbalance ratio of the dataset improves from 0.34 to 0.47 in 20 iterations of active learning. We also investigate the effectiveness of the stochastic and distributed variants of the greedy algorithm on this example. A comparison of the execution times are provided in Fig. 4[B]. For the distributed greedy method, we assume use 32 clusters, which results in each cluster having an unlabeled set of size 312. For the stochastic greedy method, we select $\epsilon = 0.01$ so that $q = 461$ samples are required with a batch-size of $B = 100$. All the algorithms produce classifiers that have an

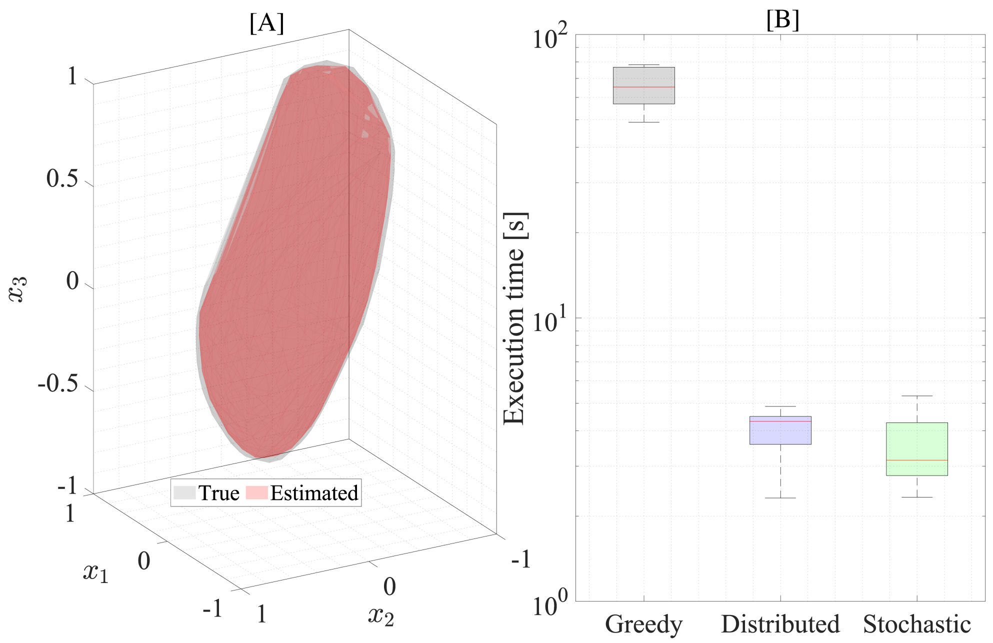

Fig. 4. [A] Inner approx. (red) of reachable set (gray) with 2000 active samples. [B] Comparison of execution times of the greedy algorithm with $|\mathcal{S}| = 2\times10^4$. The red lines within each box denote the median execution times, the shaded boxes denote the interquartile ranges, and the whiskers denote $(5, 95)$ percentiles, all measured over 100 runs.

AUC (based on a P-R curve) that is greater than 0.999. As expected, the greedy algorithm takes over an order magnitude more time to execute than the faster variants without a significant decrease in learning quality.

D. State-dependent solver selection with strong and weak oracles

We illustrate our disagreement-based active learning framework on the following linear system

$$\dot{x} = \begin{bmatrix}1 & 1\\0 & 1\end{bmatrix} x + \begin{bmatrix}0.5\\1\end{bmatrix} u,$$

which is a time-discretized version of a double integrator. Both oracles are assumed to be terminal-set constrained MPCs wherein the underlying constrained optimization problem with $T = 20$ and $q(x, u) = 0.5x^\top x + u^2$ in (5a) is solved using the ADMM algorithm. The weak oracle $\mathcal{O}_W$ is an ADMM solver that terminates with a maximum number of 10^3 iterations, and the strong oracle $\mathcal{O}_S$ is an active set solver (see Example 5).

We follow the steps in Algorithm 4. We begin with a labeled set $\mathcal{L}_{0,D}$ containing 200 samples, distributed quasi-randomly on $\mathbb{X} = \{x : \|x\|_\infty \leq 5\}$. At each labeled sample, we evaluate both $\mathcal{O}_S$ and $\mathcal{O}_W$ and label according to disagreement ($\mathcal{O}_{S,W}(x) = -1$) or agreement ($\mathcal{O}_{S,W} = +1$). We use active learning to select $N_{s,D} = 1000$ samples from 10^4 unlabeled with $B_D = 100$ and and generate outer/inner approximations of $\mathbb{X}_D$ via cost-sensitive learning where the penalty for false positives/negatives are $10\times$ higher than false negatives/positives. The results of the disagreement learning phase is shown in Fig. 5[A] using an SVM classifier. Note that there is no red sample inside $\mathcal{D}^-$ and no red sample outside $\mathcal{D}^+$, which demonstrates that the inner and outer approximations are satisfactory (based on the data available).

At the end of this phase, we have $N_s = 1500$ labeled samples in the set $\mathcal{L}_0$. We select $N_s = 500$ more active samples, with $B = 50$, and the oracle defined in (19). During the learning phase, the strong oracle was called 20% of the time, that is, of 500 samples, only 100 samples were labeled using the strong oracle, as expected. The effectiveness of the primary learner is shown in Fig. 5[B]; the learned feasible region is provided by the green line, with the true feasible region in gray shading.

VII. Conclusions

In this paper, we developed an active learning framework for estimating control-relevant sets such as reachable sets in a data-driven manner for systems that cannot be represented by closed-form analytical expressions like ODEs, DAEs, etc. Therefore, we rely on sampling and simulations to direct the set estimation problem. We pose the sample selection paradigm as a submodular maximization problem and leverage a greedy algorithm to compute solutions with guarantees. We show that for large unlabeled sets of data, one can approximate the solution of the active learning problem using randomization or distributed methods without losing significant learning performance. The potential of the approach in estimating small volumes within admissible state spaces in a data-driven manner and the critical advantage of model-free set estimation is demonstrated empirically. We also illustrate how one could use this method to select solvers for non-convex optimization problems by partitioning the feasible domain of the solvers. A theoretical open problem is to generate label complexity bounds using this active learning framework; of course, this

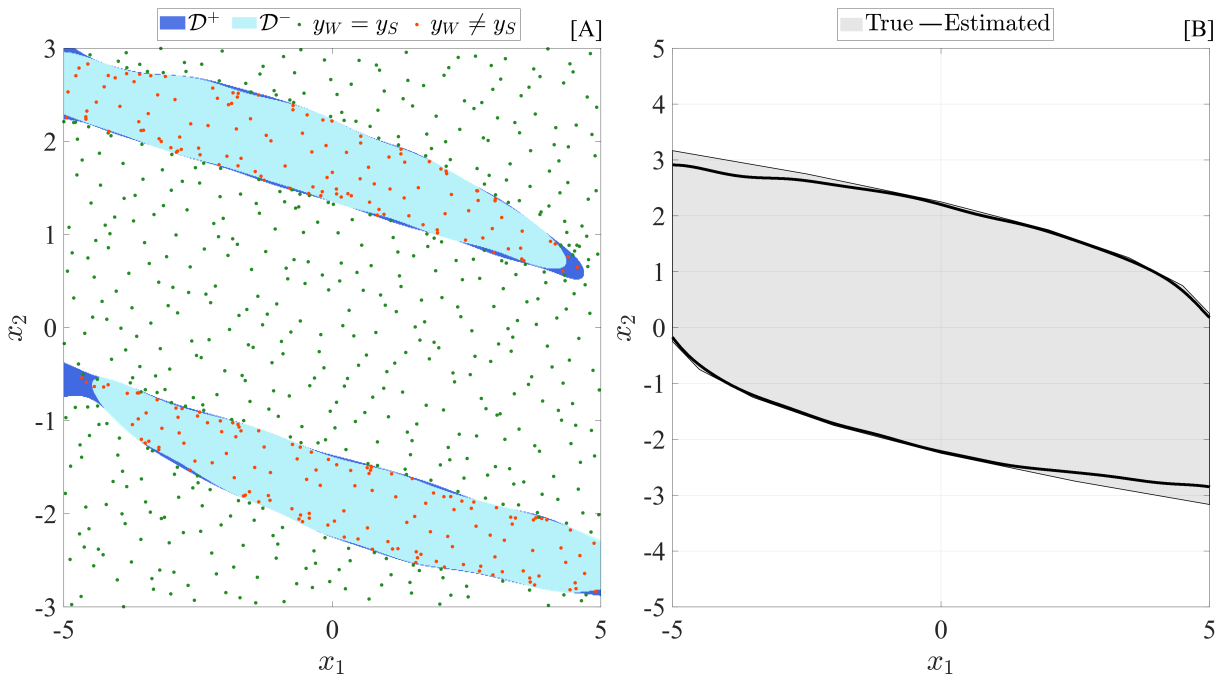

Fig. 5. [A] Illustration of $\hat{\mathcal{D}}^+$ and $\hat{\mathcal{D}}^-$ along with samples obtained from $\mathcal{L}_{0,D}$ and disagreement learning. Since the region $\hat{\mathcal{D}}^+ \setminus \hat{\mathcal{D}}^-$ is small in volume, the main learner is expected to use fewer strong oracle queries. [B] Estimate of the feasible region of the controller (gray) and its estimate (black boundary).

will require more structure on the hypothesis class and the complexity of the classification problem.

References

[1] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Proc. of the IEEE Conf. on Decision and Control*. IEEE, 2018, pp. 6059–6066.

[2] E. C. Kerrigan and J. M. Maciejowski, "Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control," in *Proc. of the IEEE Conf. on Decision and Control (CDC)*, vol. 5, 2000, pp. 4951–4956.

[3] L. Hewing, A. Carron, K. P. Wabersich, and M. N. Zeilinger, "On a correspondence between probabilistic and robust invariant sets for linear systems," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1642–1647.

[4] L. Magni, G. De Nicolao, L. Magnani, and R. Scattolini, "A stabilizing model-based predictive control algorithm for nonlinear systems," *Automatica*, vol. 37, no. 9, pp. 1351–1362, 2001.

[5] Z. Xu, H. Su, P. Shi, R. Lu, and Z.-G. Wu, "Reachable set estimation for markovian jump neural networks with time-varying delays," *IEEE Trans. on Cybernetics*, vol. 47, no. 10, pp. 3208–3217, 2016.

[6] Y. Chen, J. Lam, Y. Cui, J. Shen, and K.-W. Kwok, "Reachable set estimation and synthesis for periodic positive systems," *IEEE Trans. on Cybernetics*, 2019.

[7] S. Rozgonyi, K. Hangos, and G. Szederkényi, "Determining the domain of attraction of hybrid non-linear systems using maximal Lyapunov functions," *Kybernetika*, vol. 46, no. 1, pp. 19–37, 2010.

[8] F. Camilli, L. Grüne, and F. Wirth, "Control Lyapunov functions and Zubov's method," *SIAM Journal on Control and Optimization*, vol. 47, no. 1, pp. 301–326, 2008.

[9] A. Trofino and T. Dezuo, "LMI stability conditions for uncertain rational nonlinear systems," *International Journal of Robust and Nonlinear Control*, vol. 24, no. 18, pp. 3124–3169, 2014.

[10] J. L. Pitarch, A. Sala, and C. V. Arino, "Closed-form estimates of the domain of attraction for nonlinear systems via fuzzy-polynomial models," *IEEE Trans. on Cybernetics*, vol. 44, no. 4, pp. 526–538, 2013.

[11] G. Chesi, *Domain of attraction: analysis and control via SOS programming*. Springer Science & Business Media, 2011, vol. 415.

[12] M. Korda, D. Henrion, and C. N. Jones, "Inner approximations of the region of attraction for polynomial dynamical systems," *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 534–539, 2013.

[13] D. Han and M. Althoff, "Control synthesis for non-polynomial systems: A domain of attraction perspective," in *Proceedings of the IEEE Conference on Decision and Control*, 2015, pp. 1160–1167.

[14] D. Han and D. Panagou, "Chebyshev approximation and higher order derivatives of Lyapunov functions for estimating the domain of attraction," in *Proc. of the 56th IEEE Conference on Decision and Control (CDC)*, 2017, pp. 1181–1186.

[15] P. Polcz, T. Péni, and G. Szederkényi, "Improved algorithm for computing the domain of attraction of rational nonlinear systems," *European Journal of Control*, vol. 39, pp. 53–67, 2018.

[16] P. Polcz, T. Péni, and G. Szederkényi, "Computational method for estimating the domain of attraction of discrete-time uncertain rational systems," *European Journal of Control*, vol. 49, pp. 68–83, sep 2019.

[17] A. Chakrabarty, V. Dinh, M. J. Corless, A. E. Rundell, S. H. Żak, and G. T. Buzzard, "Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences," *IEEE Trans. on Automatic Control*, vol. 62, no. 1, pp. 135–148, 2017.

[18] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A machine learning approach for real-time reachability analysis," in *Proc. of the Int. Conf. on Intell. Robots and Systems*, 2014, pp. 2202–2208.

[19] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE Trans. on Neural Networks and Learning Systems*, no. 99, pp. 1–7, 2018.

[20] J. Zhang, X. Wu, and V. S. Shengs, "Active learning with imbalanced multiple noisy labeling," *IEEE Transactions on Cybernetics*, vol. 45, no. 5, pp. 1095–1107, 2014.

[21] A. Krause and C. Guestrin, "Nonmyopic active learning of gaussian processes: an exploration-exploitation approach," in *Proc. of the 24th International Conference on Machine Learning (ICML)*. ACM, 2007, pp. 449–456.

[22] Y. Guo, "Active instance sampling via matrix partition," in *Advances in Neural Information Processing Systems*, 2010, pp. 802–810.

[23] S. Chakraborty, V. Balasubramanian, Q. Sun, S. Panchanathan, and J. Ye, "Active batch selection via convex relaxations with guaranteed solution bounds," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 1945–1958, 2015.

[24] B. Du, Z. Wang, L. Zhang, L. Zhang, W. Liu, J. Shen, and D. Tao, "Exploring representativeness and informativeness for active learning," *IEEE Transactions on Ccybernetics*, vol. 47, no. 1, pp. 14–26, 2015.

[25] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[26] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu, "Batch mode active learning and its application to medical image classification," in *Proc. of the Int. Conf. on Machine learning (ICML)*. ACM, 2006, pp. 417–424.

[27] K. Wei, R. Iyer, and J. Bilmes, "Submodularity in data subset selection and active learning," in *International Conference on Machine Learning*, 2015, pp. 1954–1963.

[28] A. Chakrabarty, A. Raghunathan, S. Di Cairano, and C. Danielson, "Data-driven estimation of backward reachable and invariant sets for unmodeled systems via active learning," in *Proc. of the IEEE Conf. on Decision and Control*. IEEE, 2018, pp. 372–377.

[29] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, "Distributed submodular maximization," *Communications and Control Engineering*, vol. 17, no. 9783319269757, pp. 41–53, 2016.

[30] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause, "Lazier than lazy greedy," in *AAAI Conf. on AI*, 2015.

[31] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization," in *NIPS*, 2013, pp. 2049–2057.

[32] S. T. Jawaid and S. L. Smith, "Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems," *Automatica*, vol. 61, pp. 282–288, 2015.

[33] B. Gharesifard and S. L. Smith, "Distributed submodular maximization with limited information," *IEEE Trans. on Control of Network Systems*, vol. 5, no. 4, pp. 1635–1645, 2018.

[34] C. Zhang and K. Chaudhuri, "Active learning from weak and strong labelers," in *NIPS*, 2015, pp. 703–711.

[35] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4397–4404.

[36] D. J. Burns, C. Danielson, J. Zhou, and S. Di Cairano, "Reconfigurable model predictive control for multi-evaporator vapor compression systems," *IEEE Trans. on Control Systems Technology*, vol. PP, no. 99, pp. 1–17, 2017.

[37] J. H. Abel, A. Chakrabarty, and F. J. Doyle III, "Controlling biological time," in *Emerging Applications of Control and Systems Theory*. Springer, 2018, pp. 123–138.

[38] A. Chakrabarty, S. Zavitsanou, T. Sowrirajan, F. J. Doyle III, and E. Dassau, "Getting IoT-ready: The face of next generation AP systems," in *The Artificial Pancreas*. Elsevier, 2019, pp. 29–57.

[39] U. Feige, "A threshold of $ln|n|$ for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[40] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability*, 2011, pp. 71–104.

[41] A. C. Thompson, *Minkowski Geometry*. Cambridge Univ. Press, 1996.

[42] G. Still, "Lectures on parametric optimization: An introduction," *Optimization Online*, 2018.

[43] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *Journal of Machine Learning Research*, vol. 2, no. Nov, pp. 67–93, 2001.

[44] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[45] R. Barbosa, A. Ene, H. Nguyen, and J. Ward, "The power of randomization: Distributed submodular maximization on massive datasets," in *Int. Conf. on Machine Learning*, 2015, pp. 1236–1244.

[46] G. Pin *et al.*, "Approximate model predictive control laws for constrained nonlinear discrete-time systems: analysis and offline design," *Int. J. Control*, vol. 86, no. 5, pp. 804–820, 2013.

[47] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[48] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Machine Learning Research*, vol. 2, no. Nov, pp. 45–66, 2001.

[49] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network hjb approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005.