2019 American Control Conference (ACC)
Philadelphia, PA, USA, July 10-12, 2019

Training Classifiers For Feedback Control

Hasan A. Poonawala, Niklas Lauffer, and Ufuk Topcu

***Abstract*— One approach for feedback control using high dimensional and rich sensor measurements is to classify the measurement into one out of a finite set of situations, each situation corresponding to a (known) control action. This approach computes a control action without estimating the state. Such classifiers are typically learned from a finite amount of data using supervised machine learning algorithms. We model the closed-loop system resulting from control with feedback from classifier outputs as a piece-wise affine differential inclusion. We show how to train a linear classifier based on performance measures related to learning from data *and* the local stability properties of the resulting closed-loop system. The training method is based on the projected gradient descent algorithm. We demonstrate the advantage of training classifiers using control-theoretic properties on a case study involving navigation using range-based sensors.**

I. Introduction

A common situation in robotics involves using information-rich sensors, which provide high dimensional measurements, to control the state of a robot in different environments. Example of such sensors include cameras and LIDAR. Even though the available measurement is high-dimensional, the robot may often only need to identify the current situation it is in and apply a corresponding control, without explicit knowledge of the state. Obstacle avoidance using proximity sensors such as SONAR are an example of this strategy. A finite set of controls is often sufficient to achieve safe and stable operation of the robot in that environment, where each control in the set corresponds to one of the specific situations that is known to occur.

A classifier, trained using supervised learning methods, often performs the identification step. Once the measurement has been classified into one of the finite possible situations, the system uses a pre-designed control action associated with the classifier output. In many robotic systems such as for mobile robots, human expertise is sufficient to design these control actions. We refer to such a feedback control system as a classifier-in-the-loop system. Figure 1 depicts such a feedback mechanism. Several feedback systems in the literature are classifier-in-the-loop systems [1], [2], however the evaluation of their properties are almost always empirical. We seek to provide a more rigorous approach to the analysis and synthesis of classifiers used for control purposes.

This material is based upon work supported by the National Science Foundation under Grant No. 1646522 and Grant No. 1652113.

Hasan A. Poonawala is with the Department of Mechanical Engineering, University of Kentucky, Lexington, KY 40506, USA. hasan.poonawala@uky.edu

Niklas Lauffer is with the University of Texas, Austin, TX 78712, USA. nlauffer@utexas.edu

Ufuk Topcu is with the Department of Aerospace Engineering, University of Texas, Austin, TX 78712, USA. utopcu@utexas.edu

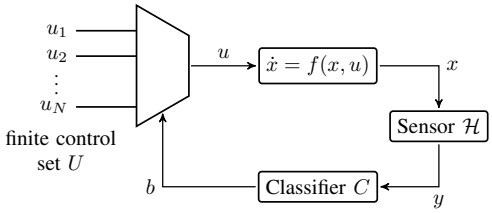


Fig. 1: A dynamical system with a classifier C in the feedback loop. The measurement y obtained by the sensor in a state x depends on the (unknown) map $\mathcal{H}$. The classifier converts the measurement into a label b that determines which control $u_i \in U$ is chosen as the control input u.

Given training data, one can use supervised learning methods [3], [4] to design a classifier that assigns one of the finite possible controls to a measurement. A common approach to supervised learning involves solution of an optimization problem. The objective function typically consists of a loss function that penalizes errors between the classifier's prediction for a measurement and the actual target value associated with that measurement in the dataset.

A low value of the loss function does not necessarily say anything about the properties of the resulting closed-loop system. We require a method to relate the closed-loop system properties with the parameters of the classifier. We wish to reformulate existing techniques for training classifiers in a way that is meaningful for their use as feedback controllers.

An important observation that permits development of the training methods we will present involves the recognition that a classifier-in-the-loop control scheme can be modeled [5] using switched [6] and/or hybrid system formalisms [7]. The classifier parameters dictate the switching (or guard) surfaces of the closed-loop system. Training the classifier is equivalent to determining the appropriate switching surface. Analysis of switched systems with variable switching surfaces is central to training of classifier-in-the-loop systems. Some methods exist to analyze or design such hybrid systems [8]–[13]. We will use methods from [11]–[13].

Contributions

This work involves three contributions. First, we show how to model the control of dynamical systems via classification using piece-wise affine differential inclusions [12]. Second, we formulate the training problem for classifiers used in control as a constrained optimization problem, and derive the corresponding constraints using Lyapunov-based stability conditions appropriate for piece-wise affine differential inclusions [11], [12]. These constraints are bilinear in the optimization variables. Our third contribution is to develop

an algorithm for solving the constrained optimization problem based on the projected gradient descent algorithm.

The work in this paper differs from [5] in that here we propose computational algorithms to design classifier-in-the-loop systems. We apply our training method for classification-based feedback control to a robot navigation problem simulated in ROS Gazebo.

II. Control-Oriented Training Of Classifiers

A classifier $C: Y \to L$ is a map that assigns a unique label $b \in L$ to an measurement $y \in Y$, where $Y \subseteq \mathbb{R}^m$ is the space of measurements. The set L is typically finite.

Consider a classifier C parametrized by a set of weights $w \in \mathbb{R}^s$ for some $s \in \mathbb{N}$. Training a classifier typically involves the solution of an optimization problem in the form

$$\min_{w \in \mathbb{R}^s} \quad l_{data}(w), \tag{1}$$

where $l_{data}: \mathbb{R}^s \to \mathbb{R}$ is a loss function for w evaluated on the data set D.

Assuming that we can compute a gradient of $l_{data}(w)$, an iterative algorithm to compute a local optimal point w^* is given by

$$w_{k+1} = w_k - \alpha_k \nabla l_{data}(w)^T \tag{2}$$

where $\nabla l_{data}(w)$ is the gradient, α_k is the learning rate, and w_k is the estimate of w^* at the k^{th} iteration.

We modify (1) to train classifiers that will be used for control in two ways. We add a term $l_{control}(w)$ to the objective function. We also add constraints on w such that all feasible solutions of the optimization problem correspond to closed-loop systems with desired behavior. Let $\mathcal{W}$ be the feasible set under these constraints. The resulting optimization problem that trains classifiers for control is

$$\min_{w \in \mathcal{W}} \quad l_{data}(w) + l_{control}(w) \tag{3}$$

Given the constraints defining $\mathcal{W}$, we can compute an optimal solution w^* using projected gradient descent. This procedure involves solution of the iterative equations

$$w'_{k+1} = w_k - \alpha_k \left(\nabla l_{data}(w) + \nabla l_{control}(w)\right)^T, \text{ and} \tag{4}$$

$$w_{k+1} = \arg\min_{w \in \mathcal{W}} \quad \|w - w'_{k+1}\|. \tag{5}$$

Figure 2 depicts the procedure.

The key problems considered in this paper are three-fold. First, how do we robustly model the closed-loop system derived from a control scheme involving classification? Second, how do we derive constraints on the classifier parameters that, if satisfied, imply desirable closed-loop properties of the modeled system? Third, when these constraints are given in the form of set $\mathcal{W}$, how do we solve the optimization problem in (5)?

The next three sections provide a solution to each of these problems. Briefly, we propose the use of piece-wise affine differential inclusions to model the closed-loop system, the use of piece-wise linear Lyapunov functions to certify closed-loop properties, and algorithms for solving biconvex optimization problems to solve (5). These solutions, together, allow us to train classifiers that yield control performance guarantees on the closed-loop system behavior. Note that some of the choices we make to solve each problem are important for being able to combine the three solutions.

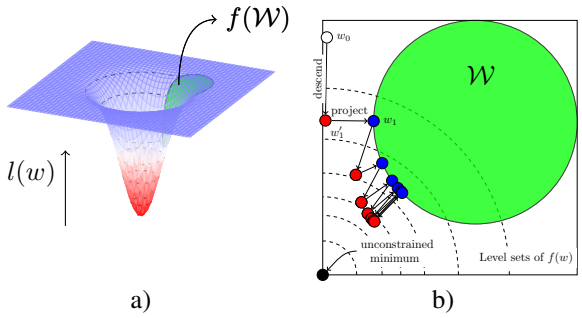


Fig. 2: The projected gradient descent algorithm. a) The surface denotes the value of the objective function, and the green patch denotes the objective values for points in the feasible region. b) The red and blue points depict the successive iterates of the algorithm. The descent step creates an infeasible point (red), which is projected onto the feasible set to obtain a feasible point (blue). In this case, the feasible iterates (blue dots) converge to the optimal solution.

III. Classifier-in-the-Loop Systems

In this section we show how to model the dynamics of a classifier-in-the loop system as a piece-wise affine differential inclusion [12] of the form

$$\dot{x}(t) \in \mathcal{A}(x(t)), \tag{6}$$

where $x \in \mathbb{R}^n$ and $\mathcal{A}: \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is a set-valued map constructed using affine functions.

A. How Do We Model Classifiers For Control?

We focus our attention on classifiers that partition the measurement space into polytopes of identically classified points. This class of classifiers is fairly large, and includes linear classifiers, rectifier networks (common in deep learning), decision trees, and nearest-neighbor classifiers. Therefore, we represent the classifier C as

$$C(y) = b_i, \quad \text{if } \bar{E}_i(w)y + \bar{e}_i(w) > 0, \tag{7}$$

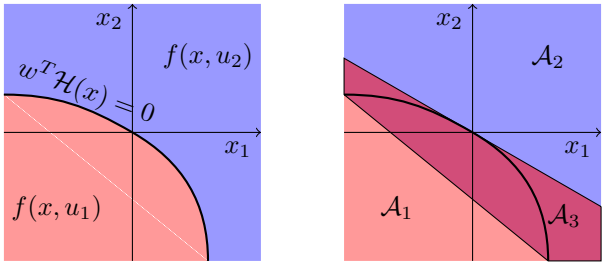
where $i \in I$, I is the index set of convex polytopes in the partition induced by the classifier. Note that non-convex polytopes are easily divided into convex polytopes.

When $L = 2$, the classifier (7) becomes

$$C(y) = \begin{cases} b_1 & , \text{ if } w_1^T y + w_0 > 0 \\ b_2 & , \text{ if } w_1^T y + w_0 < 0, \end{cases} \tag{8}$$

where $w = (w_1, w_0) \in \mathbb{R}^{m+1}$ are learned from data. In this case, the classifier parameters w in (8) directly yield the partition (equivalently, parameters $\bar{E}_i(w)$ and $\bar{e}_i(w)$) of Y. For nearest-neighbor classifiers or decision trees, the parameters $\bar{E}_i(w)$ and $\bar{e}_i(w)$ will need to be derived from the trained classifier parameters w.

When $|L| > 2$, one often constructs a classifier $C: Y \to L$ by combining multiple binary classifiers (8) in different ways. One way is to construct a decision tree, where every



a) Nonlinear closed-loop system

b) Over-approximation of Figure 3a

c) Linearization of Surface

d) Over-approximation of Figure 3c

Fig. 3: [Figure best viewed in color.] Sketch of the modeling steps that lead to a piece-wise affine differential inclusion model, such as in a). a) The (binary) classifier creates a switched dynamical system with a nonlinear switching surface. b) We can define partitions and differential inclusions to over-approximate the original system with differential inclusions $\mathcal{A}_{(\cdot)}(x)$. c) To obtain partition parameters that depend linearly on w, we can linearize $\mathcal{H}$. d) Again, we over-approximate the system to account for uncertainties.

node is a binary classifier. Another way is to train multiple classifiers parameterized as $w^j = (w_1^j, w_0^j)$, where each classifier w^j distinguishes between one of the $\binom{|L|}{2}$ possible pairs of labels from L. The partitions induced by this approach can be derived from w^j. This multi-label classification scheme is known as one-vs-one classification. Instead, we can train $|L|$ classifiers that separate each label from all other labels. This classification scheme is known as one-vs-all classification.

B. What Data Are Required?

The set D of training data generally consists of N_D triples (x^k, y^k, b^k) where x is the state of the robot, y is the measurement obtained in that state, b is the class label associated with the measurement, and k denotes the index of the triple in the dataset.

We assume that some labels $b \in L$ correspond to known control actions $u_i \in U$ that should be taken when the corresponding measurement is observed, according to the data. The classifier predicts labels for each measurement, effectively predicting a control action for each measurement.

To analyze the classifier, we also learn an approximate map $y = \hat{\mathcal{H}}(x)$. The approximation $\hat{\mathcal{H}}$ may be used for all states $x \in X$, or may be a local approximation using data corresponding to some neighborhood.

C. How Do We Model The Control?

When we associate a specific control $u_i \in U$ action to each label $b_i \in L$, the classifier C effectively assigns a control $u_i \in U$ to a measurement, where $i \in \{1, 2, \ldots, |U|\}$ (see Figure 1). That is,

$$u(t) = C(y(t)). \tag{9}$$

Recall that U is finite and $u(t)$ switches in time between its elements. Note that we obtain the control input from the measurement, *not* the state.

We approximate each vector field $f(x, u_i)$ using an affine differential inclusion $\mathcal{A}_i(x)$, given by

$$\mathcal{A}_i(x) = co\left(\{A_{ik}x + a_{ik}\}_{k \in I_{\mathcal{A}_i}}\right), \tag{10}$$

where $co(\cdot)$ is the convex hull operation, and $I_{\mathcal{A}_i}$ is the (finite) index set of the affine vector fields that define $\mathcal{A}_i(x)$. For $\mathcal{A}_i(x)$ to be a valid over-approximation of $f(x, u_i)$ over some set $S \subseteq \mathbb{R}^n$, we require that $\cup_{x \in S} f(x, u_i) \subseteq \mathcal{A}_i(x)$.

D. How Do We Derive the Closed-Loop Dynamics?

To derive a closed-loop model of the form (6), we must be able to model which labels may be assigned in a state x. We assume that there is a continuous functional relationship between the measurement and the state, at least locally in space and time, so that $y = \mathcal{H}(x)$. This assumption is reasonable for robots operating in slowly changing environments. We therefore model (9) as

$$u(t) = C\left(\mathcal{H}(x(t))\right). \tag{11}$$

We will use an approximation $\hat{\mathcal{H}}: X \to Y$ of the map $\mathcal{H}$, learned from data, to define the dynamics in a state x.

A key idea of our work is that classifier C partitions the measurement space Y, so that control (11) is effectively a state-based switching control. The dynamics switches between the vector fields $f(x, u_i)$, where $i \in \{1, \ldots, |U|\}$, which we approximate by the inclusions $\mathcal{A}_i(x)$. From equation (7), the partitions of $\mathbb{R}^n$ induced by the partitions of Y are given by inequalities of the form $\bar{E}_i(w)\hat{\mathcal{H}}(x) + \bar{e}_i(w) > 0$. These inequalities may define cells with nonlinear boundaries.

To model the closed-loop system in a way that is robust to the uncertainty in the switching surfaces, we define convex polytopic partitions where the dynamics may be a combination of the differential inclusions $\mathcal{A}_i$ that approximate the vector fields $f(x, u_i)$. Figure 3 depicts an example of this process. Figure 3a shows the case when the map $\mathcal{H}$ is nonlinear, so that the switching surface is not planar.

There are two approaches to obtaining convex partitions. The first one involves linearization of $\hat{\mathcal{H}}$ followed by over-approximation, as in Figures 3c and 3d. The linear estimate in a neighborhood of a point x_e is given by

$$\begin{aligned} y &= \hat{\mathcal{H}}(x) \\ &= \hat{\mathcal{H}}(x_e) + \frac{\partial \hat{\mathcal{H}}}{\partial x}(x - x_e) + O((x - x_e)^2) \\ &\approx Hx + h. \end{aligned} \tag{12}$$

We locally approximate the partitions by the inequalities

$$\begin{aligned} &\bar{E}_i(w)\mathcal{H}(x) + \bar{e}_i(w) \geq 0 \\ \approx &\bar{E}_i(w)(Hx + h) + \bar{e}_i(w) \geq 0 \\ = &E_i(w)x + e_i(w) \geq 0\ . \end{aligned} \tag{13}$$

The set $\mathcal{A}_3(x)$ in Figures 3b and 3d is given by $\mathcal{A}_3(x) = co(\mathcal{A}_1(x) \cup \mathcal{A}_2(x))$. The advantage of this approach is that when using binary classifiers of the form (8), the partition parameters $E_i(w)$ and $e_i(w)$ are linear in w.

A second approach is to over-approximate the nonlinear boundary in Figure 3a using polyhedral sets, as in Figure 3b. While this approach is intuitively more appealing than one involving linearization, it is harder to express the partition parameters $E_i(w)$ and $e_i(w)$ as linear functions of w.

By combining the classifier representation (7), the differential inclusion dynamics (10), and the approximations (13), we model the classifier-in-the-loop system as

$$\dot{x} \in \mathcal{A}_i(x), \quad \text{if } E_i(w)x + e_i(w) \geq 0, \tag{14}$$

which is a differential inclusion of the form $\dot{x} \in \mathcal{A}(x)$.

E. Is This Model Robust To Uncertainties?

The use of over-approximations inherently provides robustness to uncertainties and modeling errors. The caveat to this approach is that the over-approximation may exhibit far too many trajectories. Some of these trajectories may not satisfy the closed-loop properties we wish to certify for the system, while a tighter over-approximation may satisfy the control properties. Procedures to obtain such tight over-approximations are beyond the scope of this paper.

IV. Control-Oriented Constraints On Classifier Parameters

Sufficient conditions for certifying the closed-loop properties of (14) become conditions on the classifier parameters w, that is, they define $\mathcal{W}$ in (5). The properties of interest to us include practical asymptotic stability (ultimate boundedness), forward set invariance, and asymptotic stability. These properties physically correspond to low set-point or tracking errors, or to safety via boundedness of the state. We want these properties to hold for all closed-loop trajectories.

Our approach follows much of the existing work on analysis of piece-wise affine differential inclusions [11]–[13]. These papers derive sufficient conditions under different assumptions and parameterizations of the certificates (typically Lyapunov functions) for control properties. We present the conditions in terms of our chosen parameterization below. We choose polyhedral Lyapunov functions as motivated by [13], the stability conditions come from results in [12], and methods to remove quantifiers from these conditions are inspired by [11].

A partition $\mathcal{P}$ in $\mathbb{R}^n$ is a collection of subsets $\{X_i\}_{i \in I_\mathcal{P}}$; where $I_\mathcal{P}$ is an index set, $n \in \mathbb{N}$, $X_i \subseteq \mathbb{R}^n$ for each $i \in I_\mathcal{P}$, and $Int(X_i) \cap Int(X_j) = \emptyset$ for each pair $i, j \in I_\mathcal{P}$ such that $i \neq j$. We define the domain $Dom(\mathcal{P})$ of the partition as $Dom(\mathcal{P}) = \cup_{i \in I_\mathcal{P}} X_i$. We also refer to the subsets X_i in $\mathcal{P}$ as the cells of the partition. Note that this definition allows some cells in $\mathcal{P}$ to represent the boundary between other cells in $\mathcal{P}$, which is useful for handling sliding modes.

A piece-wise affine dynamical system $\Omega_\mathcal{P}$ associated with partition $\mathcal{P} = \{X_j\}_{j \in I_\mathcal{P}}$ is a collection,

$$\Omega_\mathcal{P} = \{\mathcal{A}_i(x)\}_{i \in I_\mathcal{P}} \tag{15}$$

that to each state $x \in X_i$ assigns the affine differential inclusion $\mathcal{A}_i(x) = co(\{A_{ik}x + a_{ik}\}_{k \in I_{\mathcal{A}_i}})$. Therefore,

$$\dot{x}(t) \in \mathcal{A}_i(x(t)), \text{ if } x_i(t) \in X_i. \tag{16}$$

The cell $X_i \in \mathcal{P}$ is given by

$$X_i = \{x \in \mathbb{R}^n : E_i x + e_i \geq 0\}. \tag{17}$$

We parameterize a continuous polyhedral Lyapunov function $V_\mathcal{Q}(x)$ with a partition $\mathcal{Q} = \{Z_j\}_{j \in I_\mathcal{Q}}$ and a collection of vectors $\{p_i\}_{i \in I_\mathcal{Q}}$ such that $V_\mathcal{Q}(x) = p_i^T x$, if $x \in Z_i \subseteq \mathbb{R}^n$. Each set $Z_j \in \mathcal{Q}$ is given by $Z_j = \{x \in \mathbb{R}^n : F_j x \geq 0\}$, and we assume that Z_j is pointed at the origin.

We define index sets that denote the relationship between the system $\Omega_\mathcal{P}$ and the Lyapunov function $V_\mathcal{Q}$. Let $I_{cont} \subseteq I_\mathcal{Q} \times I_\mathcal{Q}$ be the set of pairs of indices such that $Z_i \cap Z_j \neq \emptyset$. Let I_{dec} be the set of all triples (i, j, k) such that $i \in I_\mathcal{P}$, $k \in I_{\mathcal{A}_i}$, $j \in I_\mathcal{Q}$, and $X_i \cap Z_j \neq \emptyset$.

Sufficient conditions on a piece-wise differential inclusion and candidate Lyapunov function that certify the existence of the control properties under consideration are given in [12]. The result below formally states these conditions in terms of our parametrization and notation.

Lemma 1. *Let $\Omega_\mathcal{P}$ be a piece-wise affine dynamical system and $V_\mathcal{Q}$ be a candidate Lyapunov function. Let $I_\mathcal{P}$, $I_\mathcal{Q}$, I_{cont}, and I_{dec} be the index sets associated with $\Omega_\mathcal{P}$ and $V_\mathcal{Q}$. Let $Dom(\mathcal{P})$ be connected, and let $co(Dom(\mathcal{P}))$ contain the origin. Let S_{max} and S_{min} be the largest and smallest level set of $V_\mathcal{Q}(x)$ in $Dom(\mathcal{P})$.*

If the set of constraints

$$p_i = F_i^T \mu_i, \quad \forall i \in I_\mathcal{Q}, \tag{18}$$

$$\mu_i \geq \mathbf{1}, \quad \forall i \in I_\mathcal{Q}, \tag{19}$$

$$\begin{bmatrix} E_i & e_i \\ 0 & 1 \end{bmatrix}^T v_{ijk} = -\begin{bmatrix} A_{ik}^T \\ a_{ik}^T \end{bmatrix} p_j, \quad \forall (i, j, k) \in I_{dec}, \tag{20}$$

$$v_{ijk} \geq \mathbf{1}, \quad \forall (i, j, k) \in I_{dec}, \tag{21}$$

$$p_i - p_j = \lambda_{ij} f_{ij}, \quad \forall (i, j) \in I_{cont}, \text{ and} \tag{22}$$

$$\lambda_{ij} \geq 1, \quad \forall (i, j) \in I_{cont}, \tag{23}$$

is feasible, then

1) *S_{max} is invariant*
2) *S_{min} is ultimately bounded*

Furthermore, if $0 \in Dom(\mathcal{P})$, then the origin of $\Omega_\mathcal{P}$ is asymptotically stable with region of attraction S_{max}.

Proof. The proof is a straightforward combination of results from [11] and [12], which is presented in [14]. □

V. Control-Oriented Training Using Projected Gradient Descent

In this section, we present an algorithm to solve (5), given a representation of (a subset of) the set $\mathcal{W}$ in terms of (18)-(23). The constraints (18)-(23) may be infeasible, since the candidate Lyapunov function (which always exists for suitable partition $\mathcal{Q}$) may not decrease along the dynamics of $\Omega_\mathcal{P}$. To remedy this issue of infeasibility, we relax

Algorithm 1 Projection via Alternate Convex Search

Require: w'_{k+1} from (4), ϵ, β
Ensure: $w_{k+1} \in \mathcal{W}$
  $\Delta \leftarrow \infty$
  $l \leftarrow 0$ {Loop counter}
  $w(l) \leftarrow w'_{k+1}$
  while $\Delta > \epsilon$ **do**
    Compute $\Omega_{\mathcal{P}}(l)$ and $V_{\mathcal{Q}}(l)$ using $w(l)$
    $p_j^*, \mu_i^*, \lambda_{ij}^*, v_{ijk}^* \leftarrow$ Solve (24)-(30) with $w(l)$ fixed
    $w(l+1) \leftarrow$ Solve (24)-(30) with μ_i, v_{ijk}, λ_{ij} fixed as μ_i^*, v_{ijk}^*, and λ_{ij}^* respectively
    $\Delta \leftarrow \|w(l+1) - w(l)\|$
    $l \leftarrow l + 1$
  end while
  $w_{k+1} \leftarrow w(l)$
  return w_{k+1}

constraint (20). To account for this relaxation, we modify the objective function.

The following optimization problem implements (5):

$$\min_{w,p_i,u_i,v_{ijk},\lambda_{ij}} \beta\|w - w'_{k+1}\|_2 + \sum_{(i,j,k)\in I_{dec}} \|q_{ijk}\|_2 \quad (24)$$

s.t.

$$p_i = F_i^T \mu_i, \quad \forall i \in I_{\mathcal{Q}}, \quad (25)$$

$$\mu_i \geq \mathbf{1}, \quad \forall i \in I_{\mathcal{Q}}, \quad (26)$$

$$q_{ijk} = \begin{bmatrix} E_i(w) & e_i(w) \\ 0 & 1 \end{bmatrix}^T v_{ijk} + \begin{bmatrix} A_{ik}^T \\ a_{ik}^T \end{bmatrix} p_j, \quad \forall (i,j,k) \in I_{dec}, \quad (27)$$

$$v_{ijk} \geq \mathbf{1}, \quad \forall (i,j,k) \in I_{dec}, \quad (28)$$

$$p_i - p_j = \lambda_{ij} f_{ij}, \quad \forall (i,j) \in I_{cont}, \quad (29)$$

$$\lambda_{ij} \geq 1, \quad \forall (i,j) \in I_{cont}, \quad (30)$$

where q_{ijk} serve as slack variables that relax the equality constraint (20), and $\beta \in \mathbb{R}, \beta > 0$ is a weighting factor. The optimization variables include w, p_j $\forall j \in I_{\mathcal{Q}}$, v_{ijk} $\forall (i,j,k) \in I_{dec}$, and u_i $\forall i \in I_{\mathcal{Q}}$. The optimization problem (24)-(30) has the following property.

Proposition 2. *Let $\Omega_{\mathcal{P}}$ be a piece-wise affine differential inclusion and $V_{\mathcal{Q}}(x)$ be a candidate polyhedral Lyapunov function. The optimization problem* (24)-(30) *is feasible.*

Proof. The partition $\mathcal{Q}$ allows selection of vectors $\{p_j\}_{j\in I_{\mathcal{Q}}}$ such that $V_{\mathcal{Q}}(x)$ is continuous and $V_{\mathcal{Q}}(x) > 0$ for all $x \neq 0$ by construction. This property of $V_{\mathcal{Q}}(x)$ implies that constraints (25), (26), (29), and (30) are feasible. Since q_{ijk} is unconstrainted, (27) and (28) are feasible, independent of the values of the remaining optimization variables. □

The constraints (25)-(30) are bilinear in the variables of the optimization problem. Optimization problems with bilinear constraints are typically NP-hard. We use a variant of Alternate Convex Search (ACS) [15] to solve this optimization problem, given in Algorithm 1.

We begin with the classifier parameters $w'(k+1)$ obtained after a gradient descent step (4). We alternate between solving two convex optimization problems obtained by fixing a different subset of the variables in (24)-(30). We use a value of $\beta \ll 1$ to obtain solutions where the slack variables are zero. The convexity and feasibility (Proposition 2) of these problems imply that solutions exist at every iteration.

The first step of an iteration fixes w, and obtain a solution to (24)-(30) denoted by $(p_j^*, u_i^*, v_{ijk}^*, \lambda_{ij}^*)$. We then solve (24)-(30) again, however w is now a variable, and variables p_j remain variable. The variables u_i, v_{ijk}, and λ_{ij} are fixed to the corresponding values u_i^*, v_{ijk}^*, and λ_{ij}^*. The optimal solution is (p_j^{**}, w^*), and w^* is used as the fixed value of w in the next iteration of this procedure.

Note that the set I_{dec} may change as the partition $\mathcal{P}$ changes. One way to avoid needing to recompute I_{dec} at every iteration of the Alternate Convex Search is to set $\mathcal{P} = \mathcal{Q}$. This approach is implicitly taken in [11], [13].

VI. Case Study: Path Following

In our case study, we task a quadrotor equipped with an infra-red-based-scanning device to navigate a canyon-like terrain. We use the Gazebo robot simulation environment (see Figure 4), running on the Robot Operating System, to simulate this scenario. We demonstrate that the use of control constraints while training classifiers safeguards against unstable behavior.

A. Modeling

We model the quadrotor kinematics as a differential-drive like mobile robot. That is, we command the quadrotor to achieve has a forward velocity v and an angular velocity ω. The simulated quadrotor, however, possesses full inertial and rotational dynamics and implements lower-level controllers to track the commanded velocities, allowing us to abstract away those full dynamics.

The corridor/canyon defines a path in the plane. We can attach a moving coordinate frame, known as a Frenet-Serret frame, to this path, and express the dynamics of the robot within this frame (see Figure 5). The configuration of the agent in the Frenet-Serret frame is $x = (\psi, d)$, where angle ψ is the heading of the robot with respect to the path-aligned axis of the frame, and offset d is the distance between the robot's location and the origin of the frame (which lies on the path). The origin $x = 0$ corresponds to the robot being on the path with its heading aligned with the path tangent.

The robot uses three control inputs: $u_1 = \begin{bmatrix} v^* & 0 \end{bmatrix}^T$, $u_2 = \begin{bmatrix} 0 & \omega^* \end{bmatrix}^T$, and $u_3 = \begin{bmatrix} 0 & -\omega^* \end{bmatrix}^T$, where $v^* > 0$ and $\omega^* > 0$ are constants, so that $U = L = \{u_1, u_2, u_3\}$. These vectors correspond to moving forward, turning left, and turning right respectively. The dynamics under each constant input $u_i \in U$ in the local Frenet-Serret frame are given by

$$f(x, u_1) = \begin{bmatrix} \frac{v^* \rho \cos(\psi)}{1-\rho d} \\ v^* \sin(\psi) \end{bmatrix}, \; f(x, u_2) = \begin{bmatrix} \omega^* \\ 0 \end{bmatrix}, \text{ and } f(x, u_3) = \begin{bmatrix} -\omega^* \\ 0 \end{bmatrix},$$

where ρ is the (unknown) local curvature of the path.

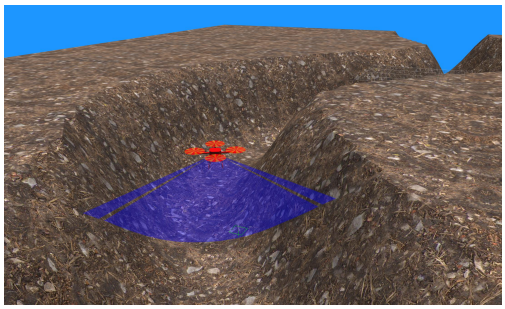


Fig. 4: [Figure best viewed in color.] Screenshot of a simulation in ROS Gazebo of a quadrotor (in red) navigating a simulated terrain. The quadrotor uses a laser scanner to sense the terrain and navigates by classifying measurements, without maintaining a state estimate. The field-of-view of the sensor is depicted by the dark blue region.

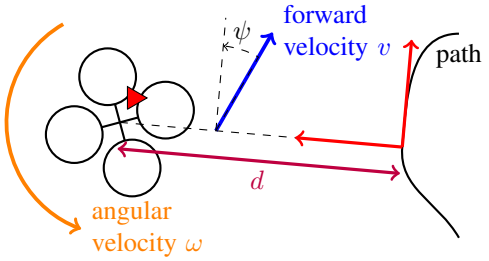


Fig. 5: A quadrotor with forward speed v, and angular velocity ω. The curved black line represents a local segment of the path that the quadrotor must follow. The local Frenet-Serret frame (red) attached to the path is also shown. The quadrotor's state consists of the offset d and angle ψ with respect to the path.

We approximate the nonlinear dynamics $f(x, u_1)$ by the affine set-valued map $\mathcal{A}_1(x)$ given by

$$\mathcal{A}_1(x) = co_{\rho \in P}\left(\begin{bmatrix} 0 & -\rho v^* \\ v^* & 0 \end{bmatrix} x + \begin{bmatrix} v^* \rho \\ 0 \end{bmatrix}\right),$$

where $P \subset \mathbb{R}$ is a closed compact set that captures the variation in curvature considered for analysis. The dynamics $f(x, u_2)$ and $f(x, u_3)$ are affine and single-valued, so that

$$\mathcal{A}_2(x) = f(x, u_2) \text{ and } \mathcal{A}_3 = f(x, u_3).$$

B. Training Data and Classification

The training data D consists of triples (x^k, y^k, b^k), where $x^k = (\psi^k, d^k)$, $d^k \in \{0.5 \text{ m}, 0 \text{ m}, -0.5 \text{ m}\}$ and $\psi^k \in \{\pi/6 \text{ rad}, 0 \text{ rad}, -\pi/6 \text{ rad}\}$. The measurement y is a vector of dimension 420. The data points x^k for which $d^k = 0$ and ψ^k is $\pi/6$ rad, 0 rad, or $-\pi/6$ rad are labeled as u_3, u_1, and u_2 respectively. We collect this data in a path that has zero curvature. The entire data set is used to estimate $\hat{\mathcal{H}}$, using polynomial regression. We take v^* and ω^* to be 0.5 m/s and 0.15 rad/s respectively.

In the rest of this section, we obtain a classifier C by training three one-vs-one classifiers w^{12}, w^{13} and w^{23} that distinguish between u_1 and u_2, u_1 and u_3, and u_2 and u_3 respectively. The loss function is

$$l_{data}(w) = \|w\|_2 + \gamma \sum_{k=1}^{N_D} \max(0, 1 - b^k y^k), \qquad (31)$$

where $\gamma > 0$ is a parameter we set as 100. The loss (31) implements a support vector machine. The class labels are then given by

$$C(y) = \begin{cases} u_2 & \text{if } (w_1^{12})^T y + w_0^{12} < 0, (w_1^{23})^T y + w_0^{23} > 0, \\ u_3 & \text{if } (w_1^{13})^T y + w_0^{13} < 0, (w_1^{23})^T y + w_0^{23} < 0, \\ u_1 & \text{otherwise.} \end{cases}$$

C. Control-Oriented Training

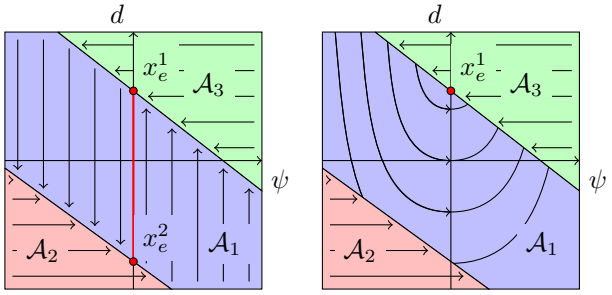
Let C_0 be the classifier obtained when training on data without control-oriented constraints. We sketch the closed-loop system due to C_0 in Figure 6. The points x_e^1 and x_e^2 are switched equilibria [6] when the curvature of the path is strictly positive and negative respectively. When the curvature is zero, every point on the line $\psi = 0$ between x_e^1 and x_e^2 is an equilibrium point. All trajectories either begin on this equilibrium set, or approach either x_e^1 or x_e^2. This analysis was presented in [5] to explain the work in [1].

To demonstrate the need for control-oriented training, we mislabel the training data, and train two sets of classifiers C_1 and C_2 using this mislabeled data. Specifically, we use the data corresponding to $(\psi^k, d^k) = (\pi/6 \text{ rad}, 0.5 \text{ m})$ as u_1, instead of the data corresponding to $(\psi^k, d^k) = (0 \text{ rad}, 0 \text{ m})$. We train C_1 by using gradient-descent to minimize (31) on the mislabeled data.

We train C_2 so that a given point x_e^1, at which $\psi = 0$ and $d > 0$, is a locally asymptotically stable equilibrium when the path curvature is positive. Similarly, we want a point x_e^2, at which $\psi = 0$ and $d < 0$, to be a locally asymptotic equilibrium point when the path curvature is negative. We achieve this training by solving the constrained optimization formulation (4) and (5) to minimize (31) on the *same data* as C_1, but subject to the following constraints. We constrain w^{13} so that $(w_1^{13})^T \mathcal{H}(x_e^1) + w_0^{13} = 0$. We use a Lyapunov function $V_{\mathcal{Q}_1}(x - x_e^1)$ to ensure that the switching between $\mathcal{A}_1$ and $\mathcal{A}_3$ renders x_e^1 to be (locally) asymptotically stable. The partition $\mathcal{Q}_1$ comprises of 16 cells depicted in Figure 7. The dynamics $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ are as in Section VI-A, where $\rho = 1$m. We solve the projection step (5) using Algorithm 1, with $\beta = 0.001$. Similarly, we train w^{12} so that x_e^2 is a switched equilibrium and $\rho = -1$m, with a different Lyapunov function $V_{\mathcal{Q}_2}(x - x_e^2)$ as proof of local asymptotic stability. We train w^{23} without any constraints, using loss function (31).

D. Results

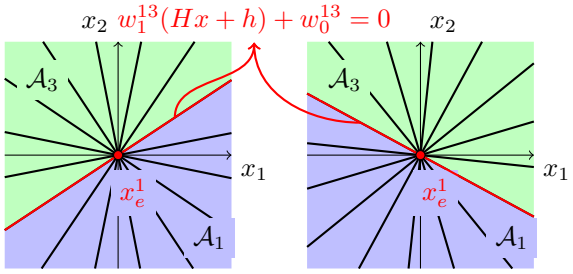
We simulate the path-following control of a quadrotor when using C_1 and C_2 (separately). Figure 8 shows the resulting trajectories, in local coordinates. We see that for the classifier C_2 trained with control constraints, the trajectories reach the set of equilibria points between x_e^1 and x_e^2. The switching surfaces are similar to those in Figure 6a. For the classifier C_1 trained without constraints on w, some trajectories move away from the origin, in fact the quadrotor crashes in the simulation. In the remaining trajectories the quadrotor is reaches a switching surface between turning left and turning right, and consequently oscillates between the two without moving along the path.



a) Closed-loop system $\Omega_{\mathcal{P}}$ when $\rho = 0$.

b) Closed-loop system $\Omega_{\mathcal{P}}$ when $\rho > 0$.

Fig. 6: Sketch of closed-loop dynamics due to classifier C_0 for path following for different curvatures. Points x_e^1 and x_e^2 are switched equilibria [6] when curvature ρ is positive and negative respectively. Their convex hull forms a line of equilibria when $\rho = 0$.



a) b)

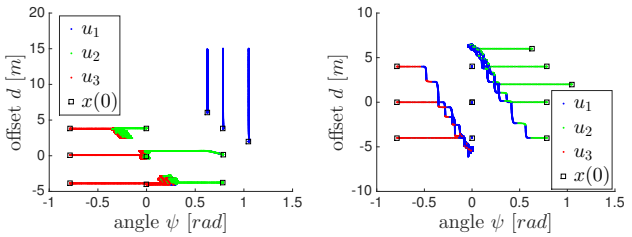
Fig. 7: Depiction of the partition $\mathcal{Q}$ that defines the Lyapunov function $V_{\mathcal{Q}}(x)$ used to train $w^{13} = (w_1^{13}, w_0^{13})$. Note that $\mathcal{P} = \mathcal{Q}$, and $\mathcal{Q}$ depends on the classifier parameters. The point x_e^1 is not asymptotically stable for the switching dynamics in a), but it is for that in b).

VII. Conclusions And Future Work

We have presented a novel training algorithm for classifiers that incorporate control-oriented constraints on the classifier parameters. We derived these constraints by modeling the closed-loop system as a piece-wise affine differential inclusion, and using polyhedral Lyapunov functions to verify desired closed-loop properties. We show the usefulness of this novel training method in a simulation of a quadrotor navigating terrain by classifying high-dimensional sensor measurements into one of three possible velocities.

While we have demonstrated the value of the proposed training method through our case study, the method presents some issues to be addressed. Our method requires us to derive a piecewise affine differential inclusion that over-approximates the effect of the classifier-in-the-loop architecture. We do not provide a systematic method to derive the tightest possible over-approximation with respect to the control properties of interest. It is possible that the over-approximation we construct does not satisfy the control properties, even though a tighter one exists that would satisfy them. Furthermore, it is unclear how the choice of the partitions for the differential inclusion and the polyhedral Lyapunov function affects the convergence of Algorithm 1. We investigate these issues in future work.

In our framework, we have assumed that the set of controls U is known, via prior knowledge or human expertise, and the measurements are labeled. In future work, we will investigate the case where the control set U needs to be determined, and/or the measurements are unlabeled.



(a) Classifier C_1 (b) Classifier C_2

Fig. 8: Trajectories of the quadrotor, in local coordinates, when following a path using classifiers trained with the same data but different methods (see Section VI-C). a) The trajectories due to classifier C_1, trained without control considerations, either exhibit oscillations due to switching between turning right and left, or the quadrotor moves away from the center of the path towards the sides, eventually crashing. b) The trajectories due to classifier C_2, trained with control constraints, approach the set of equilibria between the two points x_e^1 and x_e^2 (see Figure 6a) which exist by design of the classifier (see Section VI-C).

References

[1] A. Giusti, J. Guzzi, D. C. CireÅŸan, F. L. He, J. P. RodrÃguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, July 2016.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, Jan. 2016.

[3] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995.

[5] H. A. Poonawala and U. Topcu, "Robustness of Classifier-in-the-Loop Control Systems: A Hybrid-Systems Approach," in *IEEE Conference on Decision and Control*, 2017.

[6] A. F. Filippov and F. M. Arscott, *Differential equations with discontinuous righthand sides*, ser. Mathematics and its Applications, 1988.

[7] R. Goebel and R. Sanfelice, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[8] S. Prajna and A. Papachristodoulou, "Analysis of switched and hybrid systems - beyond piecewise quadratic methods," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 4, June 2003, pp. 2779–2784 vol.4.

[9] T. Hu, L. Ma, and Z. Lin, "Stabilization of switched systems via composite quadratic functions," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2571–2585, Dec 2008.

[10] A. Trofino, D. Assmann, C. C. Scharlau, and D. F. Coutinho, "Switching rule design for switched dynamic systems with affine vector fields," *IEEE Transactions on Automatic Control*, vol. 54, no. 9, pp. 2215–2222, Sept 2009.

[11] M. Johansson, "Piecewise linear control systems," Ph.D. dissertation, Lund University, 1999.

[12] J. Cortes, "Discontinuous dynamical systems," *IEEE Control Systems Magazine*, vol. 28, no. 3, pp. 36–73, June 2008.

[13] F. Blanchini, "Nonquadratic lyapunov functions for robust control," *Automatica*, vol. 31, no. 3, pp. 451 – 461, 1995.

[14] H. A. Poonawala, N. Lauffer, and U. Topcu, "Training Classifiers For Feedback Control," in `arXiv:1903.03688 [math.OC]`. [Online]. Available: https://arxiv.org/abs/1903.03688

[15] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: a survey and extensions," *Math Meth Oper Res*, pp. 373–407, December 2007.