A Piecewise Linear Regression and Classification Algorithm with Application to Learning and Model Predictive Control of Hybrid Systems

Alberto Bemporad

***Abstract*— This paper proposes an algorithm for solving multivariate regression and classification problems using piecewise linear predictors over a polyhedral partition of the feature space. The resulting algorithm that we call PARC (Piecewise Affine Regression and Classification) alternates between (i) solving ridge regression problems for numeric targets, softmax regression problems for categorical targets, and either softmax regression or cluster centroid computation for piecewise linear separation, and (ii) assigning the training points to different clusters on the basis of a criterion that balances prediction accuracy and piecewise-linear separability. We prove that PARC is a block-coordinate descent algorithm that optimizes a suitably constructed objective function, and that it converges in a finite number of steps to a local minimum of that function. The algorithm is used to learn hybrid numerical/categorical (HYNC) dynamical models from data that contain real and discrete labeled values. The resulting model has a piecewise linear structure that is particularly useful to formulate model predictive control problems and solve them by mixed-integer programming. A Python implementation of PARC is available at `http://cse.lab.imtlucca.it/~bemporad/parc`.**

***Index Terms*— Multivariate regression, multi-category classification, piecewise linear functions, mixed-integer programming, model predictive control, hybrid systems, piecewise affine systems**

I. Introduction

Several methods exist for solving supervised learning problems of regression and classification [1], [2]. The main goal is to estimate a model of the data generation process to *predict* at best the target value corresponding to a combination of features not seen before. However, not all methods are suitable to *optimize* on top of the estimated model, i.e., to solve a mathematical programming problem that contains the estimated model as part of the constraints and/or the objective function. An example of such a model-based optimization problem is to find the best combination of features providing a desired target, possibly under constraints on the features one can choose. In this case, the model is used as a surrogate of the underlying (and unknown) features-to-target mapping to formulate the decision problem. Applications range from derivative-free black-box optimization [3]–[7], to engineering design [8], and control engineering, in particular model predictive control [9]–[11], where actuation commands are decided in real-time by a numerical optimization algorithm based on a dynamical model of the controlled process that is learned from data [12], [13], see for instance the approach proposed recently in [14].

When optimizing over the learned model is a goal, a clear tradeoff exists between the accuracy of the model on test data and the complexity of the model, which ultimately determines the complexity of the mathematical programming problem resulting from using the model. On one extreme, we have linear regression models, which are very simple to represent as linear relations among optimization variables but have limited expressiveness. On the other extreme, random forests and other ensemble methods, k-nearest neighbors, kernel support vector machines, and other methods, can capture the underlying model very accurately but are difficult to encode in an optimization problem. Neural networks and Gaussian processes can be a good compromise between the compactness of the model and the representation of the feature-to-target relation, but are nonlinear models leading to nonconvex optimization problems that are possibly difficult to solve to global optimality.

In this paper, we advocate the use of *piecewise linear* (PWL) models as a good tradeoff between their simplicity, due to the linearity of the model on polyhedral regions of the feature-vector space, and expressiveness, due to the good approximation properties of piecewise linear functions [15]–[19]. We refer to such models with the more appropriate term *piecewise affine* (PWA), to highlight the presence of an intercept in each submodel. PWA models can be easily encoded into optimization problems by using mixed-integer linear inequalities [20]; in this way, one can optimize over them to reach a global minimum by using mixed-integer programming [21], for which excellent public domain and commercial solvers exist.

Many classical machine learning methods have an underlying PWA structure: in ridge classification, logistic (and

A. Bemporad is with the IMT School for Advanced Studies Lucca, Italy. Email: `alberto.bemporad@imtlucca.it`.
This paper was partially supported by the Italian Ministry of University and Research under the PRIN'17 project "Data-driven learning of constrained control systems", contract no. 2017J89ARP.

more generally softmax) regression, hinging hyperplane models [15], and neural networks with ReLU or leaky-ReLU activation functions, the predicted target is obtained by comparing affine functions; the predictor associated with a decision tree is a piecewise constant (PWC) function (or PWA in case of linear regression trees), over a partition of the feature-vector space in hyperboxes; the k-nearest neighbor classifier can be also expressed as a PWC function over a polyhedral partition (the comparison of squared Euclidean norms $\|x-x_i\|_2^2 \leq \|x-x_j\|_2^2$ used to determine the nearest neighbors of x is equivalent to the linear relation $2(x_j - x_i)'x \leq \|x_j\|_2^2 - \|x_i\|_2^2$), with the number of polyhedra largely growing with the number of training samples.

Different piecewise affine regression methods have been proposed in the system identification literature for getting switching linear dynamical models from data [22]–[29]. See also the survey paper [30] and the recursive PWA regression algorithms proposed in [31], [32]. Most of such methods identify a prescribed number of linear models and associate one of them to each training datapoint, therefore determining a clustering of the data. As a last step, a multicategory discrimination problem is solved to determine a function that piecewise-linearly separates the clusters [33]. For instance, the approach in [26] consists of first clustering the feature+target vectors by using a Gaussian mixture model, then using support vector classification to find hyperplanes separating adjacent clusters. In [22], the authors propose instead to cluster the vectors whose entries are the coefficients of local linear models, one model per datapoint, and then piecewise-linearly separate the clusters. In [32], K recursive least-squares problems for regression are run in parallel to cluster data in an on-line fashion, based on both the quality of fit obtained by each linear model and the proximity to the current centroids of the clusters, and finally the obtained clusters are separated by a PWL function. The aforementioned contributions to piecewise affine regression are fundamental to design model-based controllers from data that can handle *hybrid dynamics*, i.e., dynamical relations between variables of mixed real and logical nature, in particular model predictive control (MPC) laws [20].

A. Contribution

This paper proposes a general supervised learning method for regression and/or classification of multiple targets that results in a PWA predictor over a single PWA partition of the feature space in K polyhedral cells. In each polyhedron, the predictor is either affine (for numeric targets) or given by the max of affine functions, i.e., convex piecewise affine (for categorical targets). Our goal is to obtain an overall predictor that admits a simple encoding by means of binary and real variables, to be able to solve optimization problems involving the prediction function via mixed-integer linear or quadratic programming. The number K of linear predictors is limited by the tolerated complexity of the resulting mixed-integer encoding of the PWA predictor, and is therefore is a hyper-parameter of the approach deciding the trade-off between model simplicity and quality of fit.

Rather than first clustering the training data and fitting K different linear predictors, and then finding a PWL separation function to get the PWA partition, we simultaneously cluster, PWL-separate, and fit by solving a block-coordinate descent problem, similarly to the K-means algorithm [34], where we alternate between fitting models/separating clusters and reassigning training data to clusters. We call the algorithm PARC (Piecewise Affine Regression and Classification) and show that it converges in a finite number of iterations by proving that the sum of the loss functions associated with regression, classification, piecewise linear separation errors decreases at each iteration. PWL separation is obtained by solving softmax regression problems or, as a simpler alternative, by taking the Voronoi partition induced by the cluster centroids.

After showing that PARC can reconstruct an underlying PWA function from its samples, investigating the effect of K in reconstructing a nonlinear function, and comparing its performance in solving real-life regression and classification problems with respect to alternative learning methods that have a similar PWA complexity, we focus on applying the learning method to system identification of a general class of discrete-time hybrid dynamical models whose inputs, outputs, and states have a mixed numerical/categorical nature (we call them HYNC models for short). We show in detail how to efficiently encode the piecewise affine structure of HYNC models identified by PARC as a set of mixed-integer linear inequalities, so that we can formulate and evaluate model predictive controllers based on HYNC models by mixed-integer programming. We illustrate the overall data-driven hybrid control design workflow on a simple example of nonlinear switching system consisting of a cart moving between two bumpers and exchanging heat with them, based on collecting open-loop training and test data, obtaining a HYNC model by running PARC, designing a hybrid MPC controller based on that model, and then approximating the controller by a decision tree for a drastic reduction of on-line computation time.

A Python implementation of the PARC algorithm is available at `http://cse.lab.imtlucca.it/~bemporad/parc`.

B. Outline

After formulating the multivariate PWL regression and classification problem in Section II, we describe the proposed PARC algorithm and prove its convergence properties in Section III, numerically testing its behavior on different regression and classification examples. In Section IV, we show how to encode the PWA prediction function for regression and classification returned by PARC as a set of mixed-integer linear inequalities. Section V formulates a modeling framework for dynamical systems characterized by real-valued and discrete-labeled input, output, and state variables that is amenable to be identified from data by using PARC and controlled by MPC. The nonlinear switching cart example is illustrated in Section VI. Some conclusions are finally drawn in Section VII.

C. Notation and definitions

Given a finite set $\mathcal{C}$, $\operatorname{card}\mathcal{C}$ denotes its number of elements (cardinality). Given a vector $a \in \mathbb{R}^n$, $\|a\|_2$ is the Euclidean norm of a, $\|a\|_1$ its 1-norm, $[a]_i$ denotes the ith component of a. Given two vectors $a, b \in \mathbb{R}^n$, we denote by $[a = b]$ the binary quantity that is 1 if $a = b$ or 0 otherwise. Given a matrix $A \in \mathbb{R}^{m\times n}$, $\|A\|_F$ denotes the Frobenius norm of A. Given a polyhedron $P \subseteq \mathbb{R}^n$, $\mathring{P}$ denotes its interior. Given a finite set S of real numbers $\{s_1, \dots, s_K\}$ we denote by

$$\arg\min_{h\in\{1,\dots,K\}} \{s_h\} = \min\{h : \ s_h \le s_j, \ \forall j \in \{1,\dots,K\}\} \tag{1}$$

Taking the smallest index h in (1) breaks ties in case of multiple minimizers. The $\arg\max$ function of a set S is defined similarly by replacing $s_h \le s_j$ with $s_h \ge s_j$ in (1).

Definition 1: A collection $\mathcal{P}$ of sets $\{P_1, \dots, P_K\}$ is said to be a *polyhedral partition* of $\mathbb{R}^n$ if P_i is a polyhedron, $P_i \subseteq \mathbb{R}^n$, $\forall i \in \{1,\dots,K\}$, $\cup_{i=1}^K P_i = \mathbb{R}^n$, and $\mathring{P}_i \cap \mathring{P}_j = \emptyset$, $\forall i, j \in \{1,\dots,K\}$ such that $i \neq j$.

Definition 2: A function $j : \mathbb{R}^n \to \{1,\dots,K\}$ is called *integer piecewise constant* (IPWC) [35] if there exist a polyhedral partition $\mathcal{P} = \{P_1, \dots, P_K\}$ of $\mathbb{R}^n$ such that

$$j(x) = \min_h \{h \in \{1,\dots,K\} : \ x \in P_h\} \tag{2}$$

for all $x \in \mathbb{R}^n$.
The "min" in (2) prevents possible multiple definitions of $j(x)$ on overlapping boundaries $P_i \cap P_j \neq \emptyset$.

Definition 3: A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be *piecewise affine* (PWA) if there exists an IPWC function $j : \mathbb{R}^n \to \{1,\dots,K\}$ defined over a polyhedral partition $\mathcal{P}$ and K pairs (a^i, b^i), $a^i \in \mathbb{R}^{m\times n}$, $b^i \in \mathbb{R}^m$, such that

$$f(x) = a^{j(x)}x + b^{j(x)} \tag{3}$$

for all $x \in \mathbb{R}^n$. It is called *piecewise constant* if $a^i = 0$, $\forall i \in \{1,\dots,K\}$.

Definition 4: A *piecewise linear (PWL) separation* function [33] $\Phi : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$\begin{aligned} \Phi(x) &= \omega^{j(x)}x + \gamma^{j(x)} && \text{(4a)}\\ j(x) &= \arg\max_{j=1,\dots,K} \{\omega^j x + \gamma^j\} && \text{(4b)} \end{aligned}$$

where $\omega^j \in \mathbb{R}^n$, $\gamma^j \in \mathbb{R}$, $\forall j \in \{1,\dots,K\}$, and the smallest index $j(x)$ is taken in (4b) in accordance with (1).
A PWL separation function is convex [36] and PWA over the polyhedral partition $\mathcal{P} = \{P_1, \dots, P_K\}$ where

$$\begin{aligned} P_j = \{&x \in \mathbb{R}^n : \ (\omega^h - \omega^j)x \le \gamma^j - \gamma^h, \\ &\forall h \in \{1,\dots,K\}, \ h \neq j\}, \ j = 1,\dots,K \end{aligned} \tag{5}$$

II. Problem statement

We have a training dataset (x_k, y_k), $k = 1,\dots,N$, where x_k contains n_c numeric and n_d categorical features, the latter containing n_i possible values $\{v_1^i, \dots, v_{n_i}^i\}$, $i = 1,\dots,n_d$, and y_k contains m_c numeric targets and m_d categorical targets with m_i possible values $\{w_1^i, \dots, w_{m_i}^i\}$, $i = 1,\dots,m_d$. We assume that categorical features have been one-hot encoded into $n_i - 1$ binary values, so that $x_k \in \mathcal{X}$, $\mathcal{X} = \mathbb{R}^{n_c} \times \{0,1\}^{s_x}$, $s_x = \sum_{i=1}^{n_d}(n_i - 1)$. By letting $n = n_c + s_x$ we have $x_k \in \mathbb{R}^n$. Moreover, let $y_k = \left[\begin{smallmatrix} y_{ck} \\ y_{dk} \end{smallmatrix}\right]$, $y_{ck} \in \mathbb{R}^{m_c}$, $[y_{dk}]_i \in \{w_1^i, \dots, w_{m_i}^i\}$, $i = 1,\dots,m_d$, and define $\mathcal{Y} = \mathbb{R}^{m_c} \times \{w_1^1, \dots, w_{m_1}^1\} \times \dots \times \{w_1^{m_d}, \dots, w_{m_{m_d}}^{m_d}\}$, so that we have $y_k \in \mathcal{Y}$.

Several approaches exist to solve regression problems to predict the numeric components y_c and classification problems for the categorical target vector y_d. In this paper, we are interested in generalizing linear predictors for regression and classification to *piecewise linear* predictors $\hat{y} : \mathbb{R}^n \to \mathcal{Y}$ over a single *polyhedral partition* $\mathcal{P} = \{P_1, \dots, P_K\}$ of $\mathbb{R}^n$. More precisely, we want to solve the posed multivariate regression and classification problem by finding the following predictors

$$\begin{aligned} [\hat{y}_c(x)]_i &= a_i^{j(x)}x + b_i^{j(x)}, \ i = 1,\dots,m_c && \text{(6a)}\\ [\hat{y}_d(x)]_i &= w_h^i, \ h = \arg\max_{t\in I(i)} \{a_t^{j(x)}x + b_t^{j(x)}\} \\ &\quad\ i = 1,\dots,m_d && \text{(6b)} \end{aligned}$$

where $j(x)$ is defined as in (2) and the coefficient/intercept values $a^j \in \mathbb{R}^n$, $b^j \in \mathbb{R}$ define a PWA function $f : \mathbb{R}^n \to \mathbb{R}^m$ as in (3), in which $m = m_c + \sum_{i=1}^{m_d} m_i$. In (6), $I(i)$ denotes the set of indices corresponding to the ith categorical target $[y_d]_i$, $I(i) = \{t(i)+1, \dots, t(i)+m_i\}$, $t(i) = m_c + \sum_{h=1}^{i-1} m_h$. Note that subtracting the same quantity $\bar{a}x + \bar{b}$ from all the affine terms in (6b) does not change the maximizer, for any arbitrary $\bar{a} \in \mathbb{R}^n$, $\bar{b} \in \mathbb{R}$. Note also that, since according to (1) the smallest index is always taken in case ties occur when maximizing in (6b), $\hat{y}_d$ is well posed.

We emphasize that all the components of $\hat{y}(x)$ in (6) share the *same polyhedral partition* $\mathcal{P}$. A motivation for this requirement is to be able to efficiently solve optimization problems involving the resulting predictor $\hat{y}$ using mixed-integer programming, as we will detail in Section IV. Clearly, if this is not a requirement, by treating each target independently the problem can be decomposed in m_c PWA regression problems and m_d PWA classification problems.

Our goal is to jointly separate the training dataset in K clusters $\mathcal{C}_1, \dots, \mathcal{C}_K$, $\mathcal{C}_j = \{x_k : k \in J_j\}$, where $\cup_{i=1}^K J_i = \{1,\dots,N\}$, $J_i \cap J_j = \emptyset$, $\forall i, j \in \{1,\dots,K\}$, $i \neq j$, and to find optimal coefficients/intercepts a^j, b^j for (6). In particular, if the clusters were given, for each numeric target $[y_c]_i$, $i = 1,\dots,m_c$, we would solve the ridge regression problem

$$\min_{a_i^j, b_i^j} \alpha_j(\|a_i^j\|_2^2 + (b_i^j)^2) + \sum_{k\in J_j} (y_{ki} - a_i^j x_k - b_i^j)^2 \tag{7}$$

with respect to the vector $a_i^j \in \mathbb{R}^n$ of coefficients and intercept $b_i^j \in \mathbb{R}$, where $\alpha_j = \frac{\operatorname{card} J_j}{N}\alpha$ and $\alpha > 0$ is an ℓ_2-regularization parameter. For each categorical target $[y_d]_i$, $i = 1,\dots,m_d$, we would solve the regularized softmax regression problem, a.k.a. Multinomial Logistic Regression (MLR) problems [37], [38],

$$\begin{aligned} \min_{\substack{\{a_h^j, b_h^j\} \\ h \in I(i)}} \quad & \sum_{h\in I(i)} \alpha_j(\|a_h^j\|_2^2 + (b_h^j)^2) - \\ & \sum_{h=1}^{m_i} \sum_{\substack{k \in J_j : \\ [y_{dk}]_i = w_h^i}} \log \frac{e^{a_{h+t(i)}^j x_k + b_{h+t(i)}^j}}{\sum_{t\in I(i)} e^{a_t^j x_k + b_t^j}} \end{aligned} \tag{8}$$

Note that, by setting $\alpha > 0$, both (7) and (8) are strictly convex problems, and therefore their optimizers are unique. It

is well known that in the case of binary targets $[y_d]_i \in \{0,1\}$, problem (8) is equivalent to the regularized logistic regression problem

$$\min_{a_h^j, b_h^j} \alpha_j(\|a_h^j\|_2^2 + (b_h^j)^2) + \sum_{k \in J_j} \log\left(1 + e^{(1-2[y_{dk}]_i)(a_h^j x_k + b_h^j)}\right) \tag{9}$$

where $h = t(i) + 1$. Similarly, for preparing the background for what will follow in the next sections, we can rewrite (8) as

$$\begin{aligned} &\min_{\substack{\{a_h^j,\, b_h^j\} \\ h \,\in\, I(i)}} \sum_{h \in I(i)} \alpha_j(\|a_h^j\|_2^2 + (b_h^j)^2) \\ &\quad + \sum_{h=1}^{m_i} \sum_{\substack{k \,\in\, J_j\,: \\ [y_{dk}]_i \,=\, w_h^i}} \log\Big(\sum_{t \in I(i)} e^{a_t^j x_k + b_t^j}\Big) - a_{h+t(i)}^j x_k - b_{h+t(i)}^j \\ &= \min_{\substack{\{a_h^j,\, b_h^j\} \\ h \,\in\, I(i)}} \sum_{h \in I(i)} \alpha_j(\|a_h^j\|_2^2 + (b_h^j)^2) + \sum_{k \in J_j} \log\Big(\sum_{t \in I(i)} e^{a_t^j x_k + b_t^j}\Big) \\ &\quad - \sum_{h=1}^{m_i} [[y_{dk}]_i = w_h^i](a_{h+t(i)}^j x_k + b_{h+t(i)}^j) \end{aligned} \tag{10}$$

A. Piecewise linear separation

Clustering the feature vectors $\{x_k\}$ in $\mathcal{C}_1, \ldots, \mathcal{C}_K$ should be based on two goals. On the one hand, we wish to have all the data values (x_k, y_k) that can be best predicted by (a^j, b^j) in the same cluster $\mathcal{C}_j$. On the other hand, we would like the clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ to be piecewise linearly separable, i.e., that there exist a PWL separation function $\Phi : \mathbb{R}^n \to \mathbb{R}$ as in (4) such that $\mathcal{C}_i \subseteq P_i$. The above goals are usually conflicting (unless y_k depends on x_k in a piecewise linear fashion), and we will have to trade them off.

Several approaches exist to find a PWL separation function Φ of given clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$, usually attempting at minimizing the number of misclassified feature vectors x_k (i.e., $x_k \in \mathcal{C}_i$ and $x_k \notin P_i$) in case the clusters are not piecewise-linearly separable. Linear programming was proposed in [33] to solve the following problem

$$\min_{\omega,\gamma} \sum_{j=1}^{K} \sum_{\substack{h\,=\,1 \\ h\,\neq\,j}}^{K} \sum_{k:\ x_k \in \mathcal{C}_j} \frac{\max\{(\omega^h - \omega^j)x_k + \gamma^j - \gamma^j + 1, 0\}}{\operatorname{card} \mathcal{C}_j}$$

Other approaches based on the piecewise smooth optimization algorithm of [39] and averaged stochastic gradient descent [40] were described in [32]. In this paper, we use instead ℓ_2-regularized softmax regression

$$\min_{\omega,\gamma} \beta(\|\omega\|_F^2 + \|\gamma\|_2^2) + \sum_{j=1}^{K} \sum_{k:\ x_k \in \mathcal{C}_j} -\log \frac{e^{\omega^j x_k + \gamma^j}}{\sum_{i=1}^{K} e^{\omega^i x_k + \gamma^i}} \tag{11a}$$

with $\beta \geq 0$, whose solution ω, γ provides the PWL separation function (4) as

$$j(x) = \arg\max_{j=1,\ldots,K} \frac{e^{\omega^j x + \gamma^j}}{\sum_{i=1}^{K} e^{\omega^i x + \gamma^i}} = \arg\max_{j=1,\ldots,K} \omega^j x + \gamma^j \tag{11b}$$

and hence a polyhedral partition $\mathcal{P}$ of the feature vector space as in (5). Note that, as observed earlier, there are infinitely many PWL functions $\Phi(x)$ as in (4a) providing the same piecewise-constant function $j(x)$. Hence, as it is customary, one can set one pair $(\omega^i, \gamma^i) = (0, 0)$, for instance $\omega^K = 0$, $\gamma^K = 0$ (this is equivalent to dividing both the numerator and denominator in the first maximization in (11b) by $e^{\omega^K x + \gamma^K}$), and solve the reduced problem

$$\min_{\{\omega^j, \gamma^j\}_{j=1}^{K-1}} \beta(\|\omega\|_F^2 + \|\gamma\|_2^2) + \sum_{j=1}^{K} \sum_{k:\ x_k \in \mathcal{C}_j} -\log \frac{e^{\omega^j x_k + \gamma^j}}{1 + \sum_{i=1}^{K-1} e^{\omega^i x_k + \gamma^i}}$$

An alternative approach to softmax regression is to obtain $\mathcal{P}$ from the Voronoi diagram of the centroids

$$\bar{x}_j = \arg\min_x \sum_{k \in J_j} \|x_k - x\|_2^2 = \frac{1}{\operatorname{card} \mathcal{C}_j} \sum_{k \in J_j} x_k \tag{12}$$

of the clusters, inducing the PWL separation function as in (4) with

$$j(x) = \arg\min_{j=1,\ldots,K} \|x - \bar{x}_j\|_2^2 = \arg\max_{j=1,\ldots,K} \omega^j x + \gamma^j \tag{13a}$$

$$\omega^j = \bar{x}_j',\ \gamma^j = -\frac{1}{2}\|\bar{x}_j\|_2^2 \tag{13b}$$

Note that the Voronoi partitioning (13) has Kn degrees of freedom (the centroids $\bar{x}_j$), while softmax regression (11b) has $Kn + (K - n - 1)$ degrees of freedom. We finally remark that multicategory classifiers based on linear discriminant functions as in (4) are also called *linear machines* in the pattern recognition literature [41].

III. Piecewise Affine Regression and Classification Algorithm

In the previous section, we have seen how to get the coefficients a^j, b^j by ridge (7) or softmax (8) regression when the clusters $\mathcal{C}$ are given, and how to get a PWL partition of $\mathcal{C}$. The question remains on how to determine the clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$.

Let us assume that the coefficients a^j, b^j have been fixed. Following (7) and (10) we could assign each training vector x_k to the corresponding cluster $\mathcal{C}_j$ such that the following weighted sum of losses

$$\begin{aligned} V^y(a^j, b^j, x_k, y_k) &= \sum_{i=1}^{m_c} \mu_{ci}(y_{ki} - a_i^j x_k - b_i^j)^2 \\ &\quad + \sum_{i=1}^{m_d} \mu_{di} \log\Big(\sum_{t \in I(i)} e^{a_t^j x_k + b_t^j}\Big) \\ &\quad - \sum_{h=1}^{m_i} [[y_{dk}]_i = w_h^i](a_{h+t(i)}^j x_k + b_{h+t(i)}^j) \end{aligned} \tag{14}$$

is minimized, where $\mu_c \in \mathbb{R}^{m_c}$, $\mu_d \in \mathbb{R}^{m_d}$ are vectors of relative weights on fit losses.

Besides the average quality of prediction (14), we also want to consider the location of the feature vectors x_k to promote PWL separability of the resulting clusters using the

two approaches (softmax regression and Voronoi diagrams) proposed in Section II-A. Softmax regression induces the criterion

$$V_s^x(\omega^j,\gamma^j,x_k) = -\log\frac{e^{\omega^j x_k+\gamma^j}}{1+\sum_{i=1}^{K-1}e^{\omega^i x_k+\gamma^i}}$$
$$= \log\left(1+\sum_{i=1}^{K-1}e^{\omega^i x_k+\gamma^i}\right)-\omega^j x_k-\gamma^j \quad (15a)$$

for $j=1,\dots,K$, where $\omega^K=0$, $\gamma^K=0$. Note that the last logarithmic term in (15a) does not depend on j, so that it might be neglected in case V_s^x gets minimized with respect to j.

Alternatively, because of (13), Voronoi diagrams suggest penalizing the distance between x_k and the centroid $\bar{x}_j$ of the cluster

$$V_v^x(\bar{x}_j,x_k)=\|x_k-\bar{x}_j\|_2^2 \quad (15b)$$

Criteria (15a) and (15b) can be combined as follows:

$$V^x(\omega^j,\gamma^j,x_k)=\begin{cases} V_s^x(\omega^j,\gamma^j,x_k) \\ \text{if PWL partitioning (5) is used} \\ \\ V_v^x((\omega^j)',x_k)+0\cdot\gamma_j \\ \text{if Voronoi partitions (13) are used}\end{cases} \quad (15c)$$

Then, each training vector x_k is assigned to the cluster $\mathcal{C}_{j_k}$ such that

$$j_k=\arg\min_{j=1,\dots,K} V^y(a^j,b^j,x_k,y_k)+\sigma V^x(\omega^j,\gamma^j,x_k) \quad (16)$$

where $\sigma\geq 0$ is a relative weight that allows trading off between target fitting and PWL separability of the clusters. Note that, according to the definition in (1), in the case of multiple minima the optimizer j_k in (16) is always selected as the smallest index among optimal indices.

The idea described in this paper is to alternate between fitting linear predictors as in (7)–(8) and reassigning vectors to clusters as in (16), as described in Algorithm 1 that we call PARC (Piecewise Affine Regression and Classification).

The following theorem proves that indeed PARC is an algorithm, as it terminates in a finite number of steps to a local minimum of the problem of finding the K best linear predictors.

Theorem 1: Algorithm 1 converges in a finite number of steps to a local minimum of the following mixed-integer optimization problem

$$\min_{a,b,\omega,\gamma,z} \quad V(a,b,\omega,\gamma,z)$$
$$\text{s.t.} \quad \sum_{j=1}^{K} z_{kj}=1,\ k=1,\dots,N \quad (17a)$$
$$V(a,b,\omega,\gamma,z) = \sigma\beta(\|\omega\|_F^2+\|\gamma\|_2^2)$$
$$+\sum_{j=1}^{K}\sum_{k=1}^{N} z_{kj}\left(\frac{\alpha}{N}(\|a^j\|_F^2+\|b^j\|_2^2)+\right.$$
$$\left. V^y(a^j,b^j,x_k,y_k)+\sigma V^x(\omega^j,\gamma^j,x_k)\right) \quad (17b)$$

where $a^j\in\mathbb{R}^{m\times n}$, $b^j\in\mathbb{R}^m$, $\omega^j\in\mathbb{R}^{K\times n}$,$\gamma^j\in\mathbb{R}^K$, $j=1,\dots,K$, $z\in\{0,1\}^{N\times K}$, and with either $\omega^K=0$, $\gamma^K=0$, and $\beta\geq 0$ if PWL partioning (5) is used, or $\gamma^j=-\frac{1}{2}\|\omega^j\|_2^2$, $j=1,\dots,K$, and $\beta=0$ if Voronoi partions (13) are used.

Proof: We prove the theorem by showing that Algorithm 1 is a convergent block-coordinate descent algorithm (see, e.g., [42] and references therein) for problem (17), alternating between the minimization with respect to (a,b,ω,γ) and z. The proof follows arguments similar to those used to prove convergence of unsupervised learning approaches like K-means. The binary variables z_{kj} are hidden variables such that $z_{kj}=1$ if and only if the target vector y_k is predicted by $j(x_k)=j$ as in (6).

The initial clustering $\mathcal{C}_1,\dots,\mathcal{C}_K$ of $\{x_k\}$ determines the initialization of the latent variables, i.e., $z_{kj}=1$ if and only if $x_k\in\mathcal{C}_j$, or equivalently $k\in J_j$. Let us consider z fixed. Since $\sum_{j=1}^{K}\sum_{k=1}^{N} z_{kj}\left(\frac{\alpha}{N}(\|a^j\|_F^2+\|b^j\|_2^2)\right)=\sum_{j=1}^{K}\frac{\text{card }J_j}{N}\alpha(\|a^j\|_F^2+\|b^j\|_2^2)=\sum_{j=1}^{K}\sum_{i=1}^{m_c+m_d}\alpha_j(\|a_i^j\|_2^2+(b_i^j)^2)$, problem (17) becomes separable into (i) Km_c independent optimization problems of the form (7), (ii) Km_d softmax regression problems as in (8), and (iii) either a softmax regression problem as in (11a) or K optimization problems as in (12).

Let a^j, b^j, ω^j, γ^j be the solution to such problems and consider now them fixed. In this case, problem (17) becomes

$$\min_{z\in\{0,1\}^{N\times K}} \sum_{k=1}^{N}\sum_{j=1}^{K} z_{kj}(V^y(a^j,b^j,x_k,y_k)$$
$$+\sigma V^x(a^j,b^j,\omega^j,\gamma^j,x_k)) \quad (18)$$
$$\text{s.t.} \quad \sum_{j=1}^{K} z_{kj}=1,\ k=1,\dots,N$$

which is separable with respect to k into N independent binary optimization problems. The solution of (18) is given by computing j_k as in (16) and by setting $z_{j_k}=1$ and $z_j=0$ for all $j=1,\dots,K$, $j\neq j_k$.

Since the cost $V(a,b,\omega,\gamma,z)$ in (17) is monotonically nonincreasing at each iteration of Algorithm 1 and lower-bounded by zero (as all the terms in the function are nonnegative) the sequence of optimal cost values converges asymptotically. In addition, as the number of possible combinations $\{z_{kj}\}$ is finite, Algorithm 1 always terminates after a finite number of steps, since we have assumed that the smallest index j_k is always taken in (16) in case of multiple optimizers. The latter implies that no chattering between different combinations z_{kj} having the same cost V is possible. ■

Theorem 1 proved that PARC converges in a finite number of steps. Hence, a termination criterion for Step 3 of Algorithm 1 is that z does not change from the previous iteration. An additional termination criterion is to introduce a tolerance $\epsilon>0$ and stop when the optimal cost $V(a,b,\omega,\gamma,z)$ has not decreased more than ϵ with respect to the previous iteration. In this case, as the reassignment in Step 2.3.2 may have changed the z matrix, Steps 2.1.1–2.1.2 must be executed before stopping, in order to update the coefficients/intercepts (a,b) accordingly.

Note that PARC is only guaranteed to converge to a local minimum; whether this is also a global one depends on the provided initial clustering $\mathcal{C}_1,\dots,\mathcal{C}_K$, i.e., on the initial guess

Algorithm 1 PARC (Piecewise Affine Regression and Classification)

Input: Training dataset (x_k, y_k), $k = 1, \dots, N$; number K of desired linear predictors; ℓ_2-regularization parameters $\alpha > 0$, $\beta \geq 0$; fitting/separation tradeoff parameter $\sigma \geq 0$; output weight vector $\mu \in \mathbb{R}^m$, $\mu \geq 0$; initial clustering $\mathcal{C}_1, \dots, \mathcal{C}_K$ of $\{x_k\}$.

1. $i \leftarrow 1$;
2. **Repeat**
   2.1. **For all** $j = 1, \dots, K$ **do**
      2.1.1. Solve the ridge regression problem (7) for $i = 1, \dots, m_c$;
      2.1.2. Solve the softmax regression problem (8) for $i = m_c + 1, \dots, m$;
   2.2. PWL separation: either compute the cluster centroids $\omega^j = \bar{x}'_j$ (12) and set $\gamma_j = 0$, $j = 1, \dots, K$ (Voronoi partitioning), or ω^j, γ^j as in (11a) (general PWL separation);
   2.3. **For all** $k = 1, \dots, N$ **do**
      2.3.1. Evaluate j_k as in (16);
      2.3.2. Reassign x_k to cluster $\mathcal{C}_{j_k}$;
3. **Until** convergence;
4. **End**.

Output: Final number $K_f \leq K$ of clusters; coefficients a_j and intercepts b_j of linear functions, and ω^j, γ^j of PWL separation function, $j = 1, \dots, K_f$, final clusters $\mathcal{C}_1, \dots, \mathcal{C}_{K_f}$.

on z. In this paper, we initialize z by running the K-means++ algorithm [43] on the set of feature vectors $x_1, \dots, x_N$. For solving single-target regression problems, an alternative approach to get the initial clustering could be to associate to each datapoint x_k the coefficients c_k of the linear hyperplane fitting the K_n nearest neighbors of x_k (cf. [22]), for example, by setting $K_n = 2(n+1)$, and then run K-means on the set $c_1, \dots, c_n$ to get an assignment δ_k. This latter approach, however, can be sensitive to noise on measured targets and is not used in the numerical experiments reported in this paper.

As in the K-means algorithm, some clusters may become empty during the iterations, i.e., some indices j are such that $z_{kj} = 0$ for all $k = 1, \dots, N$. In this case, Step 2.1 of Algorithm 1 only loops on the indices j for which $z_{kj} = 1$ for some k. Note that the values of a^j, b^j, ω^j, and γ^j, where j is the index of an empty cluster, do not affect the value of the overall function V as their contribution is multiplied by 0 for all $k = 1, \dots, N$. Note also that some categories may disappear from the subset of samples in the cluster in the case of multi-category targets. In this case, still (8) provides a solution for the coefficients a^h_j, b^h_j corresponding to missing categories h, so that V^y in (14) remains well posed.

After the algorithm stops, clusters $\mathcal{C}_j$ containing less than $c_{\min}$ elements can be eliminated, interpreting the corresponding samples as outliers (alternatively, their elements could be reassigned to the remaining clusters). As a result, the final number K_f of clusters could be less than the starting value K allowed. We mention that after the PARC algorithm terminates, for each numeric target $[y_c]_i$ and cluster $\mathcal{C}_j$ one can further fine-tune the corresponding coefficients/intercepts a^j_i, b^j_i by choosing the ℓ_2-regularization parameter α^j in each region via leave-one-out cross-validation on the subset of datapoints contained in the cluster. In case some features or targets have very different ranges, the numeric components in x_k, y_k should be scaled.

Note that purely solving m_c ridge and m_d softmax regression problems on the entire dataset corresponds to the special case of running PARC with $K = 1$. Note also that, when $\sigma \to +\infty$, PARC will determine a PWL separation of the feature vectors, then solve m_c ridge and m_d softmax regression on each cluster. In this case, if the initial clustering $\mathcal{C}$ is determined by K-means, PARC stops after one iteration.

When the PWL separation (11a) is used, or in case of classification problems, most of the computation effort spent by PARC is due to solving softmax regression problems. In our implementation, we have used the general L-BFGS-B algorithm [44], with warm-start equal to the value obtained from the previous PARC iteration for the same set of optimization variables. Other efficient methods for solving MLR problems have been proposed in the literature, such as iteratively reweighted least squares (IRLS), that is a Newton-Raphson method [45], stochastic average gradient (SAG) descent [46], the alternating direction method of multipliers (ADMM) [47], and methods based on majorization-minimization (MM) methods [48]–[50]. Note that evaluating (14) and (15a) (as well as solving softmax regression problems) requires computing the logarithm of the sum of exponentials, see, e.g., the recent paper [51] for numerically accurate implementations.

We remark that PARC converges even if the softmax regression problem (11a) is not solved to optimality. Indeed, the proof of Theorem 1 still holds as long as the optimal cost in (11a) decreases with respect to the last computed value of ω, γ. This suggests that during intermediate PARC iterations, in case general PWL separation is used, to save computations one can avoid using tight optimization tolerances in Step 2.2. Clearly, loosening the solution of problem (11a) can impact the total number of PARC iterations; hence, there is a tradeoff to take into account.

We finally remark that Steps 2.1 and 2.3 can be parallelized for speeding computations up.

A. Predictor

After determining the coefficients a^j, b^j by running PARC, we can define the prediction functions $\hat{y}_c$, $\hat{y}_d$, and hence the overall predictor $\hat{y}$ as in (6). This clearly requires defining $j(x)$, i.e., a function that associates to any vector $x \in \mathbb{R}^n$ the corresponding predictor out of the K available. Note that the obtained clusters $\mathcal{C}_j$ may not be piecewise-linearly separable.

In principle any classification method on the dataset $\{x_k, \delta_k\}$, where $\delta_k = j$ if and only if $x_k \in \mathcal{C}_j$, can be used to define $j(x)$. For example, nearest neighbors ($j(x) = \arg\min_{k=1,\dots,N} \|x - x_k\|_2^2$), decision trees, naïve Bayes, or one-to-all neural or support vector classifiers to mention a few. In this paper, we are interested in defining $j(x)$ using a polyhedral partition $\mathcal{P} = \{P_1, \dots, P_K\}$ as stated in Section II, that is

to select $j(x)$ such that it is IPWC as defined in (2). Therefore, the natural choice is to use the values of (ω^j, γ^j) returned by PARC to define a PWL separation function by setting $j(x)$ as in (11b), which defines P_j as in (5), or, if Voronoi partitioning is used in PARC, set $j(x) = \arg\min_{j=1,\dots,K_f} \|x - \bar{x}_j\|_2^2$, which leads to polyhedral cells P_j as in (13). As the clusters $\mathcal{C}_1, \dots, \mathcal{C}_{K_f}$ may not be piecewise-linearly separable, after defining the partition $\mathcal{P} = \{P_1, \dots, P_{K_f}\}$, one can cluster the datapoints again by redefining $\mathcal{C}_j = \{x_k : \ x_k \in P_j, \ k = 1, \dots, N\}$ and then execute one last time Steps 2.1.1–2.1.2 of the PARC algorithm to get the final coefficients a, b defining the predictors $\hat{y}_c$, $\hat{y}_d$. Note that these may not be continuous functions of the feature vector x.

The number of floating point operations (flops) required to evaluate the predictor $\hat{y}(x)$ at a given x is roughly K times that of a linear predictor, as it involves K scalar products $[\omega^j \ \gamma^j] \left[\begin{smallmatrix} x \\ 1 \end{smallmatrix}\right]$ as in (11b) or (13) ($2K(n_x+1)$ flops), taking their maximum, and then evaluate a linear predictor (another $2(n_x+1)$ flops per target in case of regression (6a) and $2m_i(n_x+1)$ flops and a maximum for multi-category targets (6b)).

B. PWA form of the predictor

In the absence of categorical outputs, the PWA form of the predictor $\hat{y}$ is simply given by the polyhedra P_j computed as in (5) and the corresponding affine functions $a^j x + b^j$. In the presence of categorical outputs, the predictor $\hat{y}_d(x)$ in (6b) induces m_d separation functions $\Phi_{di} : \mathbb{R}^n \to \mathbb{R}$

$$\Phi_{di}(x) = \max_{t \in I(i)} \{a_t^{j(x)} x + b_t^{j(x)}\}, \ i = 1, \dots, m_d$$

that, in general, further subpartition the polyhedra P_j. The overall PWA partition associated with the predictor $\hat{y}(x)$ is therefore given by the intersection of the partition induced by the PWL separation function Φ associated with (ω, γ) as in (4), and by Φ_{di}, for all $i = 1, \dots, m_d$. A way to compute such an overall polyhedral partition and the corresponding affine functions is given by the simple Algorithm 2. The algorithm starts from the partition $P_1, \dots, P_K$ induced by Φ and sequentially subpartitions each polyhedron based on Φ_{di}. The final output Algorithm 2 is a PWA partition $P_1, \dots, P_{K_f}$ and the corresponding affine predictors $\hat{y}(x) = \left[\begin{smallmatrix} \hat{y}_c(x) \\ \hat{y}_d(x) \end{smallmatrix}\right] = f^j x + g^j$, $j = 1, \dots, K_f$.

C. Examples

We first test the PARC algorithm on synthetic data generated by sampling a piecewise affine function, to check how PARC can recover the function, and from a toy nonlinear function, to test the effect of the main hyper-parameters of PARC, namely K and σ. All the results have been obtained in Python 3.8.3 on an Intel Core i9-10885H CPU @2.40GHz machine. The scikit-learn package [52] is used to solve ridge and softmax regression problems, using L-BFGS to solve the nonlinear programming problem (8).

Algorithm 2 Explicit PWA predictor with categorical outputs

Input: ω, γ, a, b.

1. $K_1 \leftarrow K$; $\{P_1^1, \dots, P_{K_1}^1\} \leftarrow$ partition induced by (ω, γ) as in (5);
2. $(f_s^{j,1}, g_s^{j,1}) \leftarrow (a_s^j, b_s^j)$ for $s = 1, \dots, m_c$; $(f_s^{j,1}, g_s^{j,1}) \leftarrow (0, 0)$, for $s = m_c + 1, \dots, m_c + m_d$, $j = 1, \dots, K_1$;
3. **For all** $i = 1, \dots, m_d$ **do**:
   - 3.1. $h \leftarrow 0$;
   - 3.2. **For all** $j = 1, \dots, K_i$ **do**
     - 3.2.1. **For all** $t \in I(i)$ **do**
       - 3.2.1.1. Compute the polyhedron $P \leftarrow P_j^i \cap \{a_t^j x + b_t^j \geq a_s^j x + b_s^j, \ \forall s \in I(i), \ s \neq t\}$;
       - 3.2.1.2. **If** $P \neq \emptyset$ **then**:
         - 3.2.1.2.1 $h \leftarrow h + 1$;
         - 3.2.1.2.2 $P_h^{i+1} \leftarrow P$;
         - 3.2.1.2.3 $(f^{h,i+1}, g^{h,i+1}) \leftarrow (f^{j,i}, g^{j,i})$
         - 3.2.1.2.4 $(f_{m_c+i}^{h,i+1}, g_{m_c+i}^{h,i+1}) \leftarrow (a_t^j, b_t^j)$;
   - 3.3. $K^{i+1} \leftarrow h$;

Output: Final number $K_f = K^{m_d+1}$ of partitions and polyhedral partition $\{P_1^{m_d+1}, \dots, P_{K_f}^{m_d+1}\}$; coefficients $f^{h,m_d+1} \in \mathbb{R}^{(m_c+m_d)\times n}$ and intercepts $g^{h,m_d+1} \in \mathbb{R}^{(m_c+m_d)}$, $h = 1, \dots, K_f$.

1) Piecewise affine function: We first test whether PARC can reconstruct targets generated from the following randomly-generated PWA function

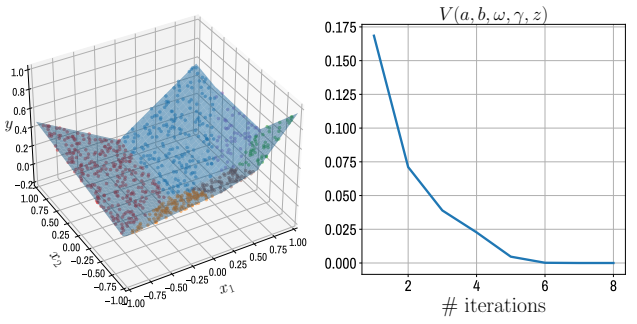
$$\begin{aligned} f(x) \ = \ \max \Bigg\{ & \begin{bmatrix} 0.8031 \\ 0.0219 \\ -0.3227 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.2458 \\ -0.5823 \\ -0.1997 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \\ & \begin{bmatrix} 0.0942 \\ -0.5617 \\ -0.1622 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.9462 \\ -0.7299 \\ -0.7141 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \\ & \begin{bmatrix} -0.4799 \\ 0.1084 \\ -0.1210 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5770 \\ 0.1574 \\ -0.1788 \end{bmatrix}' \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \Bigg\} \end{aligned} \tag{19}$$

We generate a dataset of 1000 random samples uniformly distributed in the box $[-1, 1] \times [-1, 1]$, from which we extract $N = 800$ training samples, plotted in Figures 1(a) and 1(c), and leave the remaining $N = 200$ samples for testing. Figure 1(c) shows the partition generated by the PWL function (19) as in (5).

We run PARC with $K = 6$, $\sigma = 0$, $\alpha = 10^{-5}$, PWL partitioning (11), $\beta = 10^{-3}$, $\mu = 1$, and stopping tolerance $\epsilon = 10^{-4}$ on $V(a, b, \omega, \gamma, z)$. The algorithm converges in 2.2 s after 8 iterations. The sequence of function values V is reported in Figure 1(b). The final polyhedral partition obtained by PARC is shown in Figure 1(d). In this ideal case, PARC can recover the underlying function generating the data quite well: the quality of prediction on both training and test data, measured in terms of the R^2 score

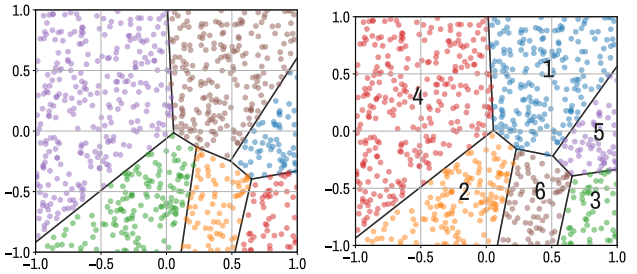
$$R^2 = 1 - \frac{\sum_{k=1}^N (y_k - \hat{y}(x_k))^2}{\sum_{k=1}^N (y_k - \frac{1}{N}\sum_{k=1}^N y_k)^2}$$

is 100%. Table I shows the results obtained when PARC is fed by feature data x_{1k}, x_{2k} and target data y_k corrupted by additive noise ε_{1k}^x, ε_{2k}^x, $\varepsilon_k^y \sim \mathcal{N}(0, \bar{\sigma}^2)$, respectively. For each value of $\bar{\sigma}$, 20 tests are run to collect statistics of the R^2 scores



(a) True PWA function (19) and dataset

(b) Cost function during PARC iterations



(c) True PWA partition induced by (19)

(d) PWA partition generated by PARC

Fig. 1. PARC algorithm for regression on training data generated by the PWA function (19).

	$\bar{\sigma}$ = 0.00	$\bar{\sigma}$ = 0.01	$\bar{\sigma}$ = 0.10	$\bar{\sigma}$ = 0.20
R^2 training (%)	100.00 (0.00)	99.84 (0.07)	95.90 (1.11)	75.95 (8.29)
R^2 test (%)	100.00 (0.00)	99.86 (0.06)	95.98 (1.24)	74.81 (8.72)

TABLE I

R^2 SCORES OBTAINED BY PARC WITH NOISY DATA $(x_{1k} + \varepsilon^x_{1k}, x_{2k} + \varepsilon^x_{2k}, y_k + \varepsilon^y_k)$, WITH $\varepsilon^x_{1k}, \varepsilon^x_{2k}, \varepsilon^y_k \sim \mathcal{N}(0, \bar{\sigma}^2)$

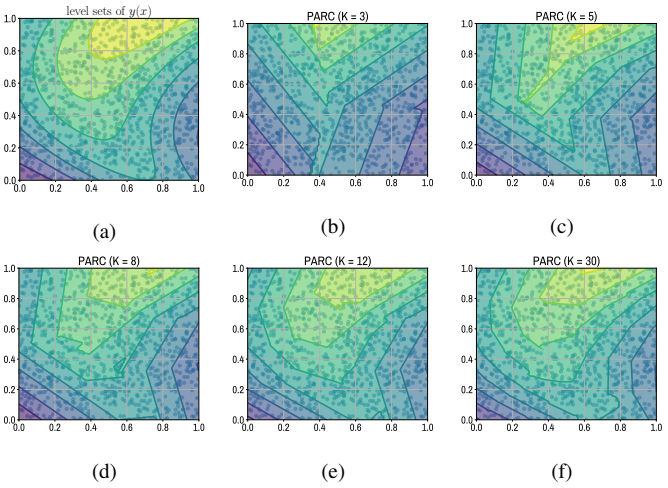
computed by comparing the new predictions $\hat{y}_k$ with respect to the original (unperturbed) data y_k. It is apparent that PARC is robust with respect to data noise.

2) Nonlinear function: We solve another simple regression example on a dataset of $N = 1000$ randomly-generated samples of the nonlinear function

$$y(x_1, x_2) = \sin\left(4x_1 - 5\left(x_2 - \frac{1}{2}\right)^2\right) + 2x_2 \qquad (20)$$

Again we use 80% of the samples as training data and the remaining 20% for testing. The function and the training dataset are shown in Figures 2(a), 3(a). We run PARC with $\sigma = 1$, $\epsilon = 10^{-4}$, $\alpha = 10^{-5}$, $\mu = 1$, PWL partitioning (11) with $\beta = 10^{-3}$, and different values of K. The level sets and training data are reported in Figure 2. The resulting piecewise linear regression functions are shown in Figure 3.

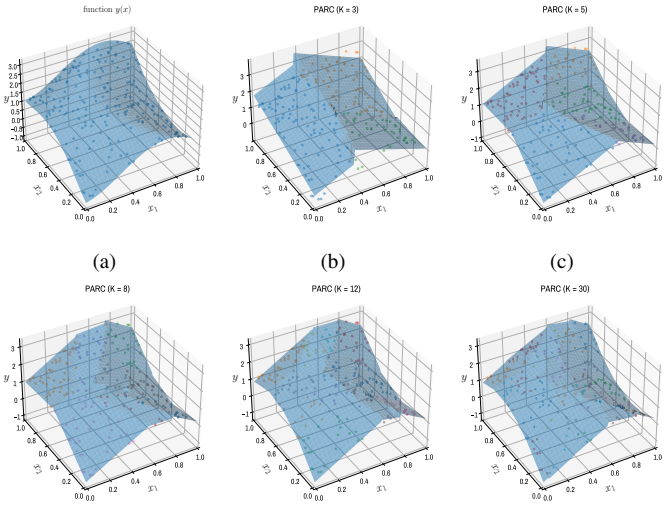
The results obtained by running PARC for different values of K, σ and the two alternative separation criteria (Voronoi partitioning and softmax regression with $\beta = 10^{-3}$) are reported in Table II (R^2-score on test data), Table III (CPU time [s] to execute PARC). The best results are usually obtained for $\sigma = 1$ using softmax regression (S) for PWL partitioning as in (11a).



(a) (b) (c)

(d) (e) (f)

Fig. 2. Training data and results of PARC for regression: nonlinear function (20).



(a) (b) (c)

(d) (e) (f)

Fig. 3. Training data and results of PARC for regression: nonlinear function (20).

Note that the case $K = 1$ corresponds to ridge regression on the entire dataset, while $\sigma = 10000$ approximates the case $\sigma \to +\infty$, corresponding to pure PWL separation + ridge regression on each cluster.

3) Real-world regression and classification datasets: We tested PARC on all the datasets from the PMLB repository [53] having between $n_x = 2$ and 20 features (before one-hot encoding categorical features), and either numeric targets and between N_{tot}=1000 and 5000 samples (see Table IV), or categorical targets with at most $m_1 = 10$ classes and between 1500 and 5000 samples (see Table VI). Further experiments from the same repository are reported in [54]. We use 80% of the available data for training, the remaining 20% for testing the resulting predictor. For comparison, we consider alternative regression and classification techniques providing piecewise linear partitions, and that therefore admit a mixed-integer encoding of the predictor of complexity similar to that of the PWA models obtained by PARC, that we will describe

	σ	$K=1$	$K=3$	$K=5$	$K=30$
(S)	0	0.548 (6.5%)	0.889 (2.6%)	0.976 (0.5%)	0.997 (0.1%)
(V)	0	0.548 (6.5%)	0.872 (3.6%)	0.970 (0.7%)	0.998 (0.1%)
(S)	0.01	0.548 (6.5%)	0.894 (2.5%)	0.976 (0.5%)	0.998 (0.1%)
(V)	0.01	0.548 (6.5%)	0.877 (3.3%)	0.969 (0.6%)	0.997 (0.1%)
(S)	1	0.548 (6.5%)	0.883 (2.8%)	0.981 (0.3%)	0.999 (0.0%)
(V)	1	0.548 (6.5%)	0.868 (3.5%)	0.970 (0.7%)	0.998 (0.1%)
(S)	100	0.548 (6.5%)	0.898 (1.6%)	0.970 (1.0%)	0.998 (0.0%)
(V)	100	0.548 (6.5%)	0.874 (4.1%)	0.967 (0.8%)	0.998 (0.1%)
(S)	10000	0.548 (6.5%)	0.816 (3.2%)	0.963 (0.8%)	0.998 (0.0%)
(V)	10000	0.548 (6.5%)	0.846 (4.4%)	0.965 (0.8%)	0.998 (0.0%)

TABLE II

PARC REGRESSION ON TARGETS FROM NONLINEAR FUNCTION (20): R^2 SCORE ON TEST DATA, MEAN (STD). PWL SEPARATION: (S) = SOFTMAX REGRESSION, (V) FOR VORONOI PARTITIONING.

	σ	$K=1$	$K=3$	$K=5$	$K=30$
(S)	0	0.12 (9.8%)	1.33 (43.6%)	1.46 (24.6%)	7.14 (17.3%)
(V)	0	0.04 (11.4%)	0.78 (36.1%)	0.77 (24.9%)	4.44 (19.5%)
(S)	0.01	0.12 (7.7%)	1.25 (40.3%)	1.48 (36.4%)	7.22 (18.7%)
(V)	0.01	0.04 (13.2%)	0.65 (41.2%)	0.82 (31.4%)	4.61 (32.8%)
(S)	1	0.12 (10.4%)	1.36 (43.8%)	1.42 (24.7%)	4.26 (14.9%)
(V)	1	0.04 (13.5%)	0.71 (38.6%)	0.78 (35.7%)	3.98 (31.1%)
(S)	100	0.12 (8.9%)	1.45 (24.2%)	1.11 (34.5%)	2.68 (22.1%)
(V)	100	0.04 (10.6%)	0.64 (44.9%)	0.69 (38.0%)	2.90 (21.0%)
(S)	10000	0.12 (9.2%)	0.24 (27.1%)	0.41 (22.6%)	2.80 (27.2%)
(V)	10000	0.04 (11.5%)	0.45 (49.7%)	0.38 (39.7%)	1.21 (61.8%)

TABLE III

PARC REGRESSION ON TARGETS FROM NONLINEAR FUNCTION (20): TRAINING TIME [S], MEAN (STD). PWL SEPARATION: (S) = SOFTMAX REGRESSION, (V) FOR VORONOI PARTITIONING.

in detail in Section IV. In particular, we consider a simple neural network (NN) with ReLU activation function with a single layer of $K_{\rm NN}=10$ neurons (in case of regression), or $K_{\rm NN}=5$ neurons (for classification), and a small decision tree (DT) with $K_{\rm DT}=10$ non-leaf nodes (for regression) or $K_{\rm DT}=5$ nodes (for classification). Note that in case of regression, NN requires $K_{\rm NN}$ binary variables to encode the ReLU activation functions as mixed-integer linear inequalities, DT requires $K_{\rm DT}$ variables, PARC K variables, while in case of classification they all require m_1 more variables. The results on regression and classification problems on training and test (reported in parentheses) data are listed in Table V and VII, respectively.

IV. MIXED-INTEGER LINEAR ENCODING

As our goal is to optimize over the estimated model $\hat{y}$, in this section we describe how to suitably encode its numeric and categorical components $\hat{y}_c$, $\hat{y}_d$ by introducing auxiliary binary variables.

A. PWA partition

In order to encode the PWA partition (5) induced by (4), one can treat (5) as a generic PWA partition and, as suggested in [20, Sect. 3.1], introduce a binary vector $\delta\in\{0,1\}^K$ and constrain

$$\begin{aligned}(\omega^i-\omega^j)x &\le \gamma^j-\gamma^i+M_{ji}(1-\delta_j)\\ &i=1,\dots,K,\ i\neq j,\ j=1,\dots,K \qquad (21a)\\ \sum_{j=1}^{K}\delta_j &= 1 \qquad (21b)\end{aligned}$$

#	dataset	$N_{\rm tot}$	n_x
1	1028_SWD	1000	21
2	1029_LEV	1000	16
3	1030_ERA	1000	51
4	529_pollen	3848	4
5	593_fri_c1_1000_10	1000	10
6	595_fri_c0_1000_10	1000	10
7	599_fri_c2_1000_5	1000	5
8	606_fri_c2_1000_10	1000	10
9	608_fri_c3_1000_10	1000	10
10	609_fri_c0_1000_5	1000	5
11	612_fri_c1_1000_5	1000	5
12	623_fri_c4_1000_10	1000	10
13	628_fri_c3_1000_5	1000	5
14	titanic	2201	5

TABLE IV

REAL-WORLD DATASETS FOR REGRESSION: DATASET NAME, TOTAL NUMBER $N_{\rm tot}$ OF SAMPLES, AND NUMBER n_x OF FEATURES BEFORE ONE-HOT ENCODING CATEGORICAL FEATURES.

#	$K=3$	$K=5$	$K=12$	ridge	NN	DT
1	0.484 (0.413)	0.515 (0.403)	0.529 (0.383)	0.441 (0.372)	0.500 (0.425)	0.388 (0.500)
2	0.600 (0.536)	0.612 (0.533)	0.623 (0.519)	0.577 (0.510)	0.567 (0.542)	0.466 (0.567)
3	0.427 (0.339)	0.427 (0.339)	0.427 (0.339)	0.427 (0.339)	0.385 (0.339)	0.347 (0.385)
4	0.794 (0.793)	0.794 (0.796)	0.796 (0.796)	0.793 (0.793)	0.791 (0.796)	0.486 (0.791)
5	0.636 (0.696)	0.755 (0.582)	0.828 (0.694)	0.306 (0.693)	0.841 (0.292)	0.751 (0.841)
6	0.805 (0.804)	0.836 (0.760)	0.893 (0.788)	0.722 (0.813)	0.899 (0.693)	0.677 (0.899)
7	0.698 (0.920)	0.849 (0.674)	0.937 (0.828)	0.312 (0.924)	0.941 (0.277)	0.791 (0.941)
8	0.617 (0.710)	0.783 (0.575)	0.855 (0.725)	0.329 (0.700)	0.857 (0.302)	0.771 (0.857)
9	0.494 (0.766)	0.854 (0.420)	0.872 (0.804)	0.305 (0.729)	0.927 (0.269)	0.748 (0.927)
10	0.821 (0.918)	0.877 (0.811)	0.934 (0.861)	0.730 (0.917)	0.895 (0.725)	0.676 (0.895)
11	0.563 (0.877)	0.750 (0.524)	0.898 (0.725)	0.264 (0.865)	0.938 (0.256)	0.746 (0.938)
12	0.675 (0.775)	0.852 (0.642)	0.887 (0.814)	0.300 (0.766)	0.933 (0.291)	0.746 (0.933)
13	0.550 (0.928)	0.907 (0.554)	0.937 (0.902)	0.268 (0.921)	0.933 (0.278)	0.738 (0.933)
14	0.295 (0.263)	0.296 (0.280)	0.279 (0.280)	0.253 (0.264)	0.289 (0.248)	0.300 (0.289)

TABLE V

REAL-WORLD DATASETS FOR REGRESSION: AVERAGE R^2 SCORE ON TRAINING (TEST) DATA OVER 20 RUNS. RIDGE REGRESSION CORRESPONDS TO $K=1$.

where ω^j,γ^j are the coefficients optimized by the PARC algorithm when the PWL separation (4) is used (with $\omega^K=0$, $\gamma^K=0$), or $\omega^j=\bar{x}'_j$ and $\gamma=-\|\bar{x}_j\|_2^2$ if Voronoi partitioning (13) is used instead. The constraint (21a) is the "big-M" reformulation of the logical constraint $[\delta_j=1]\rightarrow[x\in P_j]$, that, together with the exclusive-or (SOS-1) constraint (21b) models the constraint $[\delta_j=1]\leftrightarrow[x\in P_j]$. The values M_{ji} are upper-bounds that need to satisfy

$$M_{ji}\ge\max_{x\in\mathcal{B}}(\omega^i-\omega^j)x-\gamma^j+\gamma^i,\ i,j=1,\dots,K,\ i\neq j \quad (22)$$

where $\mathcal{B}\subset\mathbb{R}^n$ is a compact subset of features of interest. For example, given the dataset $\{x_k\}_{k=1}^N$ of features, we can set $\mathcal{B}$ as a box containing all the sample feature vectors so that the values M_{ij} in (22) can be easily computed by solving $K(K-1)$ linear programs. A simpler way to estimate the

#	dataset	$N_{\rm tot}$	n_x	m_1
1	car	1728	15	4
2	churn	5000	21	2
3	GAMETES_E**0.1H	1600	40	2
4	GAMETES_E**0.4H	1600	38	2
5	GAMETES_E**0.2H	1600	40	2
6	GAMETES_H**_50	1600	39	2
7	GAMETES_H**_75	1600	39	2
8	led7	3200	7	10
9	mfeat_morphological	2000	7	10
10	segmentation	2310	21	7
11	wine_quality_red	1599	11	6
12	wine_quality_white	4898	11	7

TABLE VI

REAL-WORLD DATASETS FOR CLASSIFICATION: DATASET NAME, TOTAL NUMBER $N_{\rm tot}$ OF SAMPLES, NUMBER n_x OF FEATURES BEFORE ONE-HOT ENCODING CATEGORICAL FEATURES, NUMBER m_1 OF TARGET CLASSES.

#	$K=2$	$K=3$	$K=5$	softmax	NN	DT
1	0.960 (0.937)	0.969 (0.944)	0.982 (0.949)	0.947 (0.934)	0.969 (0.952)	0.835 (0.983)
2	0.907 (0.901)	0.898 (0.888)	0.896 (0.882)	0.867 (0.862)	0.934 (0.921)	0.941 (0.937)
3	0.644 (0.540)	0.644 (0.508)	0.700 (0.532)	0.562 (0.478)	0.751 (0.622)	0.566 (0.578)
4	0.694 (0.620)	0.722 (0.631)	0.800 (0.676)	0.551 (0.467)	0.836 (0.752)	0.544 (0.703)
5	0.616 (0.511)	0.641 (0.512)	0.686 (0.507)	0.578 (0.514)	0.675 (0.509)	0.566 (0.528)
6	0.625 (0.521)	0.643 (0.525)	0.700 (0.542)	0.552 (0.490)	0.769 (0.648)	0.569 (0.632)
7	0.629 (0.510)	0.695 (0.612)	0.732 (0.584)	0.564 (0.487)	0.792 (0.682)	0.563 (0.641)
8	0.750 (0.732)	0.751 (0.731)	0.752 (0.731)	0.747 (0.735)	0.730 (0.724)	0.690 (0.732)
9	0.768 (0.739)	0.768 (0.736)	0.775 (0.739)	0.758 (0.737)	0.744 (0.733)	0.716 (0.747)
10	0.974 (0.954)	0.980 (0.954)	0.980 (0.955)	0.967 (0.958)	0.966 (0.954)	0.941 (0.966)
11	0.622 (0.584)	0.635 (0.588)	0.661 (0.583)	0.607 (0.594)	0.619 (0.595)	0.618 (0.599)
12	0.552 (0.542)	0.555 (0.537)	0.570 (0.536)	0.540 (0.538)	0.555 (0.542)	0.545 (0.552)

TABLE VII

REAL-WORLD DATASETS FOR CLASSIFICATION: AVERAGE ACCURACY SCORE ON TRAINING (TEST) DATA OVER 20 RUNS. SOFTMAX REGRESSION CORRESPONDS TO $K=1$.

values M_{ji} is given by the following lemma [55, Lemma 1]:

Lemma 1: Let $\mathcal{B}=\{x\in\mathbb{R}^n:\ x_{\min}\le x\le x_{\max}\}$ and $v\in\mathbb{R}^n$. Let $v^+=\max\{v,0\}$, $v^-=\max\{-v,0\}$. Then

$$\sum_{i=1}^n v_i^+x_{\min,i}-v_i^-x_{\max,i}\le v'x\le\sum_{i=1}^n v_i^+x_{\max,i}-v_i^-x_{\min,i} \tag{23}$$

Proof: Since $x_{\min,i}\le x_i\le x_{\max,i}$ and $v=v^+-v^-$, we get

$$v'x=\sum_{i=1}^n v_ix_i=\sum_{i=1}^n(v_i^+-v_i^-)x_i\le\sum_{i=1}^n v_i^+x_{\max,i}-v_i^-x_{\min,i}$$

and similarly $v'x\ge\sum_{i=1}^n v_i^+x_{\min,i}-v_i^-x_{\max,i}$. ■

By applying Lemma 1 for $v=\omega^i-\omega^j$, (22) is satisfied by setting

$$\begin{aligned}M_{ji}=\ &\gamma^i-\gamma^j+\sum_{h=1}^n\max\{\omega_h^i-\omega_h^j,0\}x_{\max,h}\\&-\max\{\omega_h^j-\omega_h^i,0\}x_{\min,h}\end{aligned} \tag{24}$$

for all $i,j=1,\dots,K,\ i\ne j$.

The mixed-integer encoding of the partition as in (21a) requires $K(K-1)$ linear inequalities. Next Lemma 2 proves that the partition induced by the PWL separation function (4), or alternatively (13), can be encoded instead by $2K$ linear inequalities, i.e., by a number of inequalities that is linear rather than quadratic in K.

Lemma 2: Consider the PWL separation function (4) or (13). Let

$$\begin{aligned}M_j&=\sum_{h=1}^n\max\{\omega_h^j,0\}x_{\max,h}-\max\{-\omega_h^j,0\}x_{\min,h}+\gamma^j\\m_j&=\sum_{h=1}^n\max\{\omega_h^j,0\}x_{\min,h}-\max\{-\omega_h^j,0\}x_{\max,h}+\gamma^j\end{aligned} \tag{25}$$

and set $m_\Phi=\max_{j=1,\dots,K}\{m_j\}$, $M_\Phi=\min_{j=1,\dots,K}\{M_j\}$. Let $\delta_j\in\{0,1\}$, $j=1,\dots,K$, and $\epsilon\in\mathbb{R}$ be auxiliary variables. Then, for all $x_{\min}\le x\le x_{\max}$, if the following inequalities

$$\begin{aligned}&\epsilon\ge(\omega^j)'x+\gamma^j+(m_\Phi-M_j)\delta_j,\ j=1,\dots,K\\&\epsilon\le(\omega^j)'x+\gamma^j+(M_\Phi-m_j)(1-\delta_j),\ j=1,\dots,K\\&\textstyle\sum_{j=1}^K\delta_j=1\end{aligned} \tag{26}$$

are satisfied for $\delta_j=1$, then $x\in P_j$.

Proof: Assume $\delta_j=1$. Then $\delta_i=0$ for all $i\ne j$. From (26) we get

$$(\omega^i)'x+\gamma^i\le\epsilon\le(\omega^j)'x+\gamma^j,\ \forall i\ne j$$

which corresponds to $x\in P_j$. We show next that the remaining inequalities are all redundant. Since $\epsilon\ge(\omega^i)'x+\gamma^i$ for all $i\ne j$ then by Lemma 1 $\epsilon\ge m_i$ for all $i\ne j$, and hence $\epsilon\ge m_\Phi\ge(\omega^j)'x+\gamma^j+(m_\Phi-M_j)$. In addition, $\epsilon\le(\omega^i)'x+\gamma^i$ for all $i\ne j$, and hence by Lemma 1 $\epsilon\le M_i$ for all $i\ne j$, so that $\epsilon\le M_\Phi\le(\omega^j)'x+\gamma^j+(M_\Phi-m_j)$ already holds. ■

Note that the mixed-integer encoding introduced above does not guarantee that when x belongs to more than one neighboring polyhedra the minimum index j in (2) is selected by a numerical solver. This can easily remedied for example by adding a very small penalty on $\sum_{j=1}^K j\delta_j$ to break ties.

B. Numeric targets

Having encoded the PWL partition, as suggested in [20, Sect. 3.1] one can define the ith predictor by imposing

$$[\hat y_c(x)]_i=\sum_{j=1}^K p_{ji} \tag{27}$$

where $p_{ji}\in\mathbb{R}$ are auxiliary optimization variables representing the product $p_{ji}=\delta_j(a_i^jx+b_i^j)$. This is modeled by the

following mixed-integer linear inequalities

$$\begin{array}{rcl} p_{ji} & \leq & a_i^j x + b_i^j - M_{ji}^{c-}(1-\delta_j) \\ p_{ji} & \geq & a_i^j x + b_i^j - M_{ji}^{c+}(1-\delta_j) \\ p_{ji} & \leq & M_{ji}^{c+}\delta_j \\ p_{ji} & \geq & M_{ji}^{c-}\delta_j \end{array} \tag{28}$$

The coefficients M_{ji}^{c-}, M_{ji}^{c+} need to satisfy $M_{ji}^{c-} \leq \min_{x\in\mathcal{B}} a_i^j x + b_i^j \leq \max_{x\in\mathcal{B}} a_i^j x + b_i^j \leq M_{ji}^{c+}$ and can be obtained by linear programming or, more simply, by applying Lemma 1.

Next Lemma 3 provides a more efficient formulation that does not require introducing the auxiliary variables p_{ji}.

Lemma 3: Let δ_j, $j = 1,\dots,K$, satisfy (21b). Then the condition $[\delta_j = 1] \rightarrow [[\hat{y}_c(x)]_i = a_i^j x + b_i^j]$ is equivalent to imposing the mixed-integer linear inequalities

$$\begin{array}{rcl} [\hat{y}_c(x)]_i - a_i^j x - b_i^j & \leq & (-M_{ji}^{c-} + \max_j\{M_{ji}^{c+}\})(1-\delta_j) \\ [\hat{y}_c(x)]_i - a_i^j x - b_i^j & \geq & (-M_{ji}^{c+} + \min_j\{M_{ji}^{c-}\})(1-\delta_j) \end{array} \tag{29}$$

Proof: Clearly (29) is equivalent to $[\hat{y}_c(x)]_i = a_i^j x + b_i^j$ when $\delta_j = 1$. For $\delta_j = 0$, by (21b) $[\hat{y}_c(x)]_i = a_i^h x + b_i^h$ for some $h \neq j$, and hence $[\hat{y}_c(x)]_i \leq M_{hi}^{c+} \leq \max_j\{M_{ji}^{c+}\} \leq \max_j\{M_{ji}^{c+}\} + a_i^j x + b_i^j - M_{ji}^{c-}$, which makes the first inequality in (29) redundant. The redundancy of the second inequality in (29) when $\delta_j = 0$ can be proved similarly. ■

C. Categorical targets

Regarding the m_d classifiers $\hat{y}_{di}$, to model the "$\arg\max$" in (6b) we further introduce s_y binary variables $\nu_{ih} \in \{0,1\}$, $h = 1,\dots,m_i$, $i = 1,\dots,m_d$, satisfying the following big-M constraints

$$(a_h^j - a_t^j)x \geq b_t^j - b_h^j - M_{ht}^d(2 - \nu_{ih} - \delta_j)$$
$$\forall h, t \in I(i),\ h \neq t,\ j = 1,\dots,K \tag{30a}$$
$$\sum_{h\in I(i)} \nu_{ih} = 1,\ i = 1,\dots,m_d \tag{30b}$$

where the coefficients M_{ht}^d must satisfy $M_{ht}^d \geq \max_{j=1,\dots,K}\{\max_{x\in\mathcal{B}}(a_t^j - a_h^j)x + b_t^j - b_h^j\}$. Note that the constraints in (30a) become redundant when $\delta_j = 0$ or $\nu_{ih} = 0$ and lead to $a_h^j x + b_h^j \geq a_t^j + b_t^j$ for all $t \in I(i)$, $t \neq h$, when $\nu_{ih} = \delta_j = 1$, which is the binary equivalent of $[\hat{y}_d(x)]_i = w_h^i$ for $x \in P_j$. Then, the ith classifier is given by

$$[\hat{y}_d(x)]_i = \sum_{h=1}^{m_i} w_h^i \nu_{ih} \tag{30c}$$

Note that in case of binary targets $[\hat{y}_d(x)]_i \in \{0,1\}$, we do not need to introduce the auxiliary variables ν_{ih}, $h = 1,2$, as we can simply replace $\nu_{i1} = [\hat{y}_d(x)]_i$, $\nu_{i2} = 1 - [\hat{y}_d(x)]_i$ in (30a). Moreover, a preprocessing similar to the analysis performed by Algorithm 2 can be carried out to identify logic constraints $[\delta_j = 1] \rightarrow [\nu_{ih} = 0]$ whenever the polyhedron $P_{jih} = \emptyset$, where

$$\begin{array}{rcl} P_{jih} & = & \{x:\ (\omega^i - \omega^j)'x \leq \gamma^j - \gamma^i,\ \forall i \neq j, \\ & & (a_t^j - a_h^j)x \leq b_h^j - b_t^j,\ \forall t \in I(i),\ t \neq h\} \end{array}$$

for $j \in \{1,\dots,K\}$, $i \in \{1,\dots,m_d\}$, and $h \in I(i)$.

Note also that, in alternative to using (30), one can use the explicit characterization provided by Algorithm 2 and encode the PWC functions $[\hat{y}_d]_i : \mathbb{R}^n \rightarrow \{w_1^i,\dots,w_{m_i}^i\}$ as they were numeric targets.

In conclusion, we have provided a mixed-integer linear reformulation of the predictors $\hat{y}_c, \hat{y}_d$ returned by the PARC algorithm as in (6). This enables solving optimization problems involving the estimated model, possibly under additional linear and logical constraints (also reformulated as mixed-integer linear inequalities [20]) on features and targets, to global optimality. Note that if one is interested in solving an optimization problem based on a more refined nonlinear predictor $\hat{y}_{NL}$, for example, a feedforward neural network trained on the same dataset, the solution obtained by solving the problem based on the PWA model identified by PARC can be used to warm-start the nonlinear programming solver based on $\hat{y}_{NL}$, which would give better chances to find a global minimizer.

V. Hybrid Numerical/Categorical Systems

In the previous sections, we have introduced an algorithm to solve a multivariate regression and classification problem in PWA form, and showed how to encode the resulting predictor by means of mixed-integer linear inequalities for its exploitation in optimization. In this section, we introduce a general class of hybrid dynamical systems with mixed numeric (real) and categorical (discrete) inputs, outputs, and states, that we call in short hybrid numerical/categorical (HYNC) systems. Our goal is to leverage on the PARC algorithm to identify a discrete-time hybrid dynamical model from data and solve model predictive problem based on that model by mixed integer linear or quadratic programming.

Let $u = \left[\begin{smallmatrix} u_c \\ u_d \end{smallmatrix}\right]$ denote the input of the system, consisting of $\bar{m}_c$ real components, $u_c \in \mathbb{R}^{\bar{m}_c}$, and $\bar{m}_d$ categorical components $[u_d]_i \in \{\bar{u}_1^i,\dots,\bar{u}_{\bar{m}_i}^i\}$; let $\zeta = \left[\begin{smallmatrix} \zeta_c \\ \zeta_d \end{smallmatrix}\right]$ be the output of the system, $\zeta_c \in \mathbb{R}^{p_c}$, and p_d categorical components $[\zeta_d]_i \in \{\bar{\zeta}_1^i,\dots,\bar{\zeta}_{p_i}^i\}$; let $\xi = \left[\begin{smallmatrix} \xi_c \\ \xi_d \end{smallmatrix}\right]$ be the state of the system, $\xi_c \in \mathbb{R}^{\bar{n}_c}$, and $\bar{n}_d$ categorical components $[\xi_d]_i \in \{\bar{\xi}_1^i,\dots,\bar{\xi}_{\bar{n}_i}^i\}$; let k denote the discrete-time index. We denote by $e_u(u_d)$ the one-hot binary encoding of u_d, $e_{ui} : \{\bar{u}_1^i,\dots,\bar{u}_{\bar{m}_{ui}}^i\} \rightarrow \{0,1\}^{\bar{m}_i}$, $i = 1,\dots,\bar{m}_d$, and similarly $e_\xi(\xi_d)$ the one-hot encoding of ξ_d, $e_{\xi i} : \{\bar{\xi}_1^i,\dots,\bar{\xi}_{\bar{n}_i}^i\} \rightarrow \{0,1\}^{\bar{n}_i}$, $i = 1,\dots,\bar{n}_d$.

Consider the state-space hybrid dynamical model

$$\left\{\begin{array}{rcl} \xi(k+1) & = & f(\xi(k),u(k)) \\ \zeta(k) & = & g(\xi(k),u(k)) \end{array}\right. \tag{31}$$

where the first $\bar{n}_c$ components of f, corresponding to numeric states, and the first p_c components of g, corresponding to numeric outputs, are PWA functions defined in accordance with Definition 3 as

$$\begin{array}{rcl} f_i(\xi(k),u(k)) & = & a_{\xi i}^{j(x(k))} x(k) + b_{\xi i}^{j(x(k))},\ i = 1,\dots,\bar{n}_d \\ g_i(\xi(k),u(k)) & = & a_{\zeta i}^{j(x(k))} x(k) + b_{\zeta i}^{j(x(k))},\ i = 1,\dots,p_d \\ x(k) & = & \left[\begin{smallmatrix} \xi_c(k) \\ e_\xi(\xi_d(k)) \\ u_c(k) \\ e_u(u_d(k)) \end{smallmatrix}\right] \end{array} \tag{32}$$

and $j(x)$ is the index of the partition, defined as in (4b). In accordance with (6b), the remaining categorical components of f and g are defined by the piecewise linear classifiers

$$f_i = \bar{\xi}_h^i,\ h = \arg\max_{t\in I_x(i)} \{a_{\xi t}^{j(x(k))} + b_{\xi t}^{j(x(k))}\},\ i = 1,\dots,\bar{n}_d$$
$$g_i = \bar{y}_h^i,\ h = \arg\max_{t\in I_\zeta(i)} \{a_{\zeta t}^{j(x(k))} + b_{\zeta t}^{j(x(k))}\},\ i = 1,\dots,p_d$$

where $I_x(i)$ denotes the set of indices corresponding to the ith categorical next state $[\xi_d(k+1)]_i$, $I_x(i) = \{t_x(i)+1,\dots,t_x(i)+\bar{n}_i\}$, $t_x(i) = \bar{n}_c + \sum_{h=1}^{i-1}\bar{n}_h$, and similarly $I_\zeta(i)$ is the set of indices corresponding to the ith categorical output $[\zeta_d(k)]_i$, $I_\zeta(i) = \{t_\zeta(i)+1,\dots,t_\zeta(i)+p_i\}$, $t_\zeta(i) = p_c + \sum_{h=1}^{i-1} p_h$.

Given a dataset $\xi(0), u(0), \zeta(0), \dots, \xi(N-1), u(N-1), \zeta(N-1), \xi(N)$, the HYNC model (31) can be estimated using the PARC algorithm, with feature vector $x(k)$ and target vector $y(k) = \begin{bmatrix}\xi(k+1)\\ \zeta(k)\end{bmatrix}$. Note that in the special case categorical inputs, states, and outputs are all binary, the HYNC model (31) corresponds to a discrete-time PWA model [20], [56]–[59] with state $\xi \in \mathbb{R}^{\bar{n}_c}\times\{0,1\}^{\bar{n}_d}$, input $u \in \mathbb{R}^{\bar{m}_c}\times\{0,1\}^{\bar{m}_d}$, and output $\zeta \in \mathbb{R}^{p_c}\times\{0,1\}^{p_d}$.

If only input/output data are available, in alternative to the state-space form (31) we can consider the following hybrid numerical/categorical autoregressive with exogenous inputs (HYNCARX) form

$$\zeta(k) = h(\zeta(k-1),\dots,\zeta(k-n_a),u(k-1),\dots,u(k-n_b)) \tag{33}$$

where n_a, n_b are integers determining the order of the model (it is immediate to extend (33) to more flexible model structures in which $\zeta_i(k)$ depends on past outputs $\zeta_j(k-1),\dots,\zeta_j(k-n_{aij})$ and past inputs $u_j(k-n_{ki}-1),\dots,u_j(k-n_{ki}-n_{bij})$, $n_{ki}\in\mathbb{Z}$, $n_{ki}\geq -1$). In (33), h is a PWA function defined similarly to f and g in (32), with $x(k)$ composed by $\zeta_c(k-1)$, $e_\zeta(\zeta_d(k-1))$, …, $\zeta_c(k-n_a)$, $e_\zeta(\zeta_d(k-n_a))$, …, $u_c(k-1)$, $e_u(u_d(k-1))$, …, $u_c(k-n_b)$, $e_u(u_d(k-n_b))$, where $e_\zeta(\zeta_d)$ is the one-hot binary encoding of ζ_d. Clearly, model (33) can be estimated using the PARC algorithm, with feature vector $x(k)$ and target vector $y(k)=\zeta(k)$.

A. Hybrid MPC based on HYNC models

A natural way to control the system that has generated the dataset used by PARC is to design a model predictive controller based on the resulting state-space HYNC model (31). Consider the following finite-time optimal control problem

$$\begin{aligned}\min_z\ & \sum_{t=0}^{T-1}\ell_{k+t}(\zeta_t,\xi_t,u_t)\\ \text{s.t.}\ & \xi_{t+1} = f(\xi_t,u_t),\quad \xi_0=\xi(k)\\ & \zeta_t = g(\xi_t,u_t)\\ & \text{arbitrary linear and logical constraints on}\\ & \quad u_{c,0}, e_u(u_{d,0}),\dots,u_{c,N-1},e_u(u_{d,N-1}),\\ & \quad \xi_{c,0},e_\xi(\xi_{d,0}),\dots,\xi_{d,N},e_\xi(\xi_{d,N}),\\ & \quad \zeta_{c,0},e_\zeta(\zeta_{d,0}),\dots,\zeta_{c,N-1},e_\zeta(\zeta_{d,N-1})\end{aligned} \tag{34}$$

where $z = \begin{bmatrix}u_0' & \xi_1' & \mu_0 & \dots & u_{T-1}' & \xi_T' & \mu_{T-1}'\end{bmatrix}'$ is the optimization vector, μ_t collects the auxiliary binary variables ϵ_t, δ_t, and $\nu_{ih,t}$ required to encode the model identified by PARC as described in (26), (29), (30), and ℓ_{k+t} is any cost function that can be encoded as a linear or quadratic function of z and possibly other auxiliary optimization variables, for example $\ell_{k+t}(\zeta_t,\xi_t,u_t) = \|\Lambda_c(\zeta_{ct}-r_c(k+t))\|_1 + \|\Lambda_d(e_\zeta(\zeta_{dt}) - e_\zeta(r_d(k+t)))\|_1$, where $r(k) = \begin{bmatrix}r_c(k)\\ r_d(k)\end{bmatrix}$ is the output reference to track and Λ_c, Λ_d are (in general, diagonal) weight matrices of appropriate dimensions. Note that any constraint that admits an encoding as mixed-integer linear inequalities, such as box and general polyhedral constraints, logical constraints (truth of Boolean functions involving $e_\xi(\xi_{k+t}), e_u(u_{k+t}), e_\zeta(u_{k+t})$ and $\delta(k)$), and mixed linear/logical constraint (such as implications between numeric and one-hot encoded and other binary variables), possibly involving the addition of auxiliary real and binary variables, can be included in (34), see e.g. [60], [61].

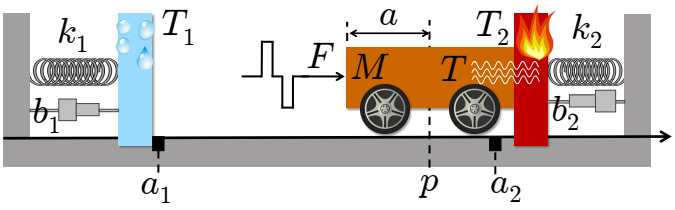


Fig. 4. Toy example: cart system with bumpers and heat exchange

VI. Hybrid Model Learning and Control Example

To illustrate the use of PARC for hybrid system identification and MPC based on the learned model, we consider the toy system depicted in Figure 4. This consists of a cart moving longitudinally between two bumpers. We denote, respectively, by M, p, and $\dot{p}$ the mass, position, and velocity of the cart, and by a half its size; the coefficient b determines the viscous friction force, T, Θ, R are the temperature, heat capacity, and thermal resistance when exchanging heat with the environment, respectively, of the cart, and T_0 is the environmental temperature. The bumpers are characterized by their mass M_i, coefficients k_i and b_i determining, respectively, the spring and viscous forces, and the position a_i at which they stop, $i=1,2$. We assume that the continuous dynamics are nonlinear, as the viscous friction force depends also on the squared velocity and the spring force also on the cubic deflection, as detailed below in (37). We assume that the bumpers have constant temperatures T_1, T_2, and that heat transfer occurs when the cart and the bumper get in contact, with thermal resistance R_1, R_2, respectively.

The cart is actuated by a switch $u\in\{-1,0,1\}$, a categorical input that triggers the force $F = \bar{F}_0 u$ on the cart. We consider the following categorical output

$$\zeta_d(t) = \begin{cases}\text{green} & \text{if } T < T_A\\ \text{yellow} & \text{if } T_A \leq T \leq T_B\\ \text{red} & \text{if } T > T_B\end{cases} \tag{35}$$

We describe the dynamics of the system by the following

Quantity	Value	Description	Units
M	1	cart mass	kg
b	0.1	cart viscous friction	N/(m/s)
a	0.1	half cart length	m
Θ	5	cart heat capacity	J/K
R	10	cart thermal resistance	K/W
T_0	25 + 273.15	ambient temperature	K
M_1, M_2	0.2	bumper mass	kg
b_1, b_2	0.2	bumper viscous friction	N/(m/s)
k_1, k_2	1	bumper spring constant	N/m
a_1	1	bumper #1 position	m
a_2	3	bumper #2 position	m
R_1, R_2	1	cart-bumper thermal resistance	K/W
T_1	10 + 273.15	temperature of bumper #1	K
T_2	50 + 273.15	temperature of bumper #2	K
T_A	30 + 273.15	threshold temperature	K
T_B	35 + 273.15	threshold temperature	K
$\bar{F}$	0.5	input force	N

TABLE VIII
NUMERICAL VALUES USED IN THE TOY EXAMPLE

switching nonlinear differential equations

$$\begin{cases} \ddot{p} &= \frac{1}{M+M_1}(F - \varphi_b(\dot{p}, b_1 + b) - \varphi_a(p - a - a_1, k_1)) \\ \dot{T} &= \frac{1}{\Theta}\left(\frac{T_0 - T}{R} + \frac{T_1 - T}{R_1}\right), \quad \text{if } p - a \le a_1 \end{cases}$$
$$\begin{cases} \ddot{p} &= \frac{1}{M+M_2}(F - \varphi_b(\dot{p}, b_2 + b) - \varphi_a(p + a - a_2, k_2)) \\ \dot{T} &= \frac{1}{\Theta}\left(\frac{T_0 - T}{R} + \frac{T_2 - T}{R_2}\right), \quad \text{if } p + a \ge a_2 \end{cases}$$
$$\begin{cases} \ddot{p} &= \frac{1}{M}(F - \varphi_b(\dot{p}, b)) \\ \dot{T} &= \frac{T_0 - T}{R\Theta}, \quad \text{otherwise} \end{cases} \tag{36}$$

where

$$\varphi_a(\Delta p, k) = k\Delta p + \frac{k}{5}\Delta p^3, \quad \varphi_b(\dot{p}, b) = b\dot{p} + \frac{b}{5}|\dot{p}|\dot{p} \tag{37}$$

The state of the system is $\xi = [p\ \dot{p}\ T]'$. The numerical values used to simulate the system are reported in Table VIII. Note that all such values are supposed *totally unknown*, they are only used to simulate the system.

Starting from the initial condition $p(0) = 2$, $\dot{p}(0) = 0$, $T_0 = 25 + 273.15$, we simulate the system for 2,000 s by numerically integrating (36) using Adams/BDF method with automatic stiffness detection and switching[1], with the input u allowed to switch with probability 5% every $T_s = 0.5$ s. When a switch is allowed, either $u = \pm 1$ (with probability $\frac{10}{3}$%) or $u = 0$. The simulated trajectories are sampled every T_s time units to create the training dataset, depicted in Figure 5, which therefore consists of 4,000 samples. The system is simulated again from $p(0) = 1.5$, $\dot{p}(0) = 0$, $T_0 = 40 + 273.15$ for 500 time units to create a test dataset.

We run PARC with $\sigma = 1$ and different values of K, using a PWL separation function to create the partition, taking $\begin{bmatrix} \xi(kT_s) \\ u(kT_s) \end{bmatrix}$ as the feature vector, and $[p((k+1)T_s), \dot{p}((k+1)T_s), T((k+1)T_s), \zeta_d(kT_s)]'$ as the target vector. The possible values of the categorical output ζ_d are mapped to 0=green, 1=yellow, and 2=red. The temperature T is converted to Celsius and divided by 10 for better numerical scaling during training. Table IX shows the R^2 and accuracy scores $a = \frac{1}{N}\sum_{k=1}^{N}[\hat{\zeta}_{dk} = \zeta_{dk}]$ for $K = 3, 5, 7$, the CPU time taken by PARC, and its number of iterations. The table shows the scores obtained on both one-step-ahead and open-loop prediction errors. The latter are computed by simulating the HYNC model produced by PARC, as shown in Figure 6 on test scenarios when $K = 5$.

[1]We use the `integrate.solve_ivp` function with method 'LSODA' of the `scipy` package (`scipy.org`).

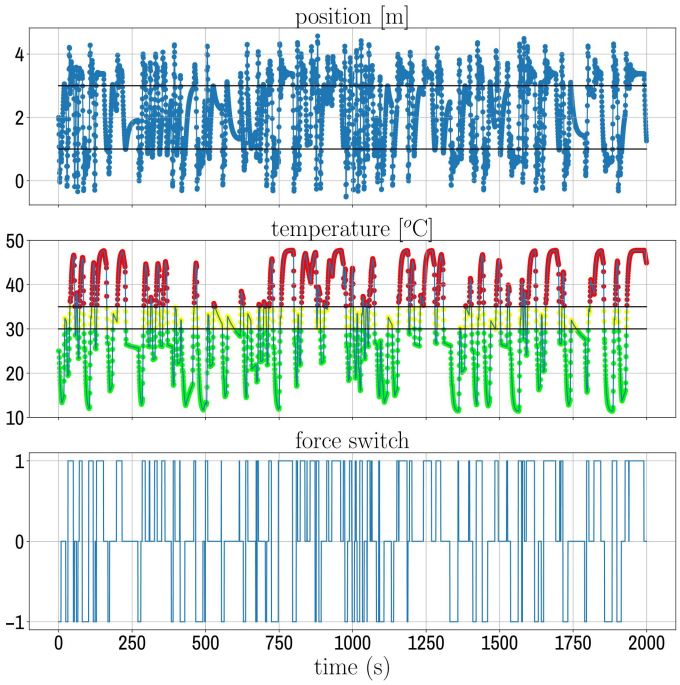


Fig. 5. Toy example: training dataset

K		$R^2(p)$	$R^2(\dot{p})$	$R^2(T)$	$a(\zeta)$	time	iters
3	1S	1.0000	0.9986	0.9998	0.9960	14.1 s	7
	OL	0.8410	0.8170	0.5367	0.7080		
5	1S	1.0000	0.9987	0.9996	0.9930	34.3 s	10
	OL	0.9500	0.9502	0.9123	0.7550		
7	1S	1.0000	0.9987	0.9998	0.9780	60.6 s	14
	OL	0.9386	0.9360	0.9243	0.8120		

TABLE IX
R^2 SCORES AND ACCURACY OBTAINED BY PARC ON THE TOY EXAMPLE ON TEST DATA FOR DIFFERENT VALUES OF K (1S = ONE-STEP AHEAD PREDICTION ERRORS, OL = OPEN-LOOP PREDICTION ERRORS).

Next, we design an MPC controller whose aim is to keep the cart in the yellow zone, possibly avoiding applying nonzero forces. To this end, we consider the following MPC formulation

$$\begin{aligned} \min \quad & \sum_{t=0}^{T-1} |\zeta_{d,t} - 1| + \rho|u| \\ \text{s.t.} \quad & \text{identified HYNC model} \\ & u \in \{-1, 0, 1\} \end{aligned} \tag{38}$$

with prediction horizon $T = 9$ and $\rho = 0.25$. Problem (38) is converted to a mixed-integer linear programming (MILP) problem by introducing a slack variable $\epsilon_t \ge \pm(\zeta_{d,t} - 1)$ per prediction step, and by encoding the identified HYNC model as described in Section IV. The hybrid MPC controller based on the model identified by PARC with $K = 5$ is simulated in closed-loop with the continuous-time simulator of the system

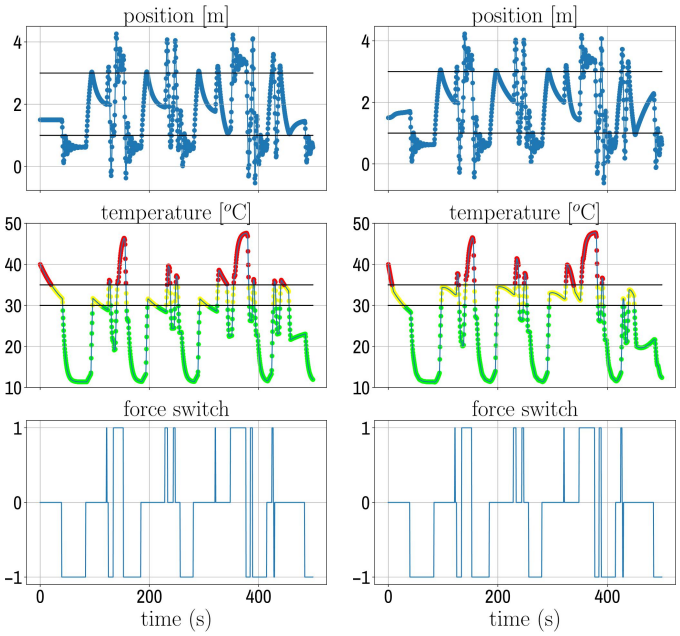


Fig. 6. Toy example: test dataset (left) and open-loop simulation of the identified HYNC model with $\boldsymbol{K = 5}$ (right).

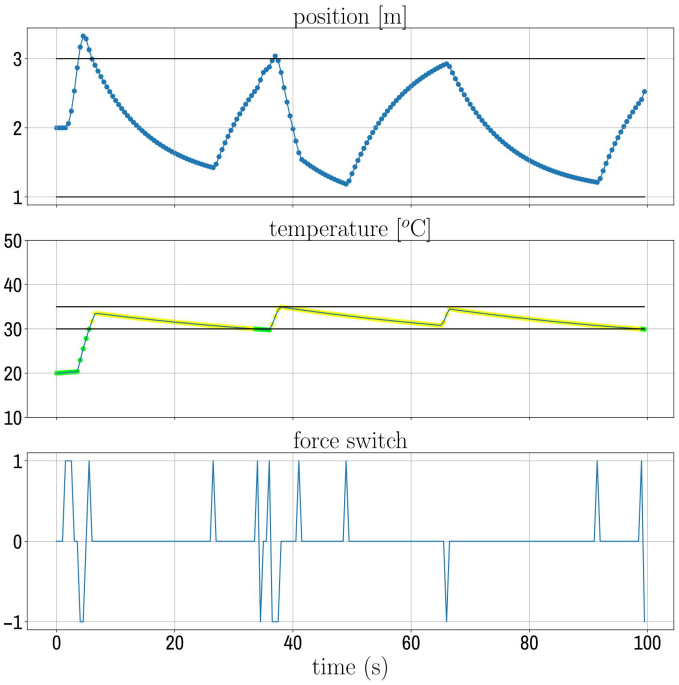


Fig. 7. Toy example: hybrid closed-loop MPC results with prediction model obtained by PARC with $\boldsymbol{K = 5}$.

for 100 s. The obtained trajectories are depicted in Figure 7.

The MILP formulation of (38) is evaluated in Python via the PuLP package [2] and CPLEX 12.10 [62], with a total CPU time ranging between 0.39 and 2.5 s (0.74 s on average) per controller execution. For comparison, the CPU time for the MPC controller based on $K = 3$ takes 0.15÷0.29 s, 0.18 s on average, with slightly worse closed-loop performance.

In order to reduce computation time, we get an approximate explicit form of the controller by generating 10,000 random samples of the state vector uniformly distributed in $\{[p\ \dot{p}\ T] : 0 \leq p \leq 5,\ 1.5 \leq \dot{p} \leq 1.5,\ 20 \leq T \leq 40\}$ and train a decision-tree (DT) classifier with maximum depth of 20 using the `tree.DecisionTreeClassifier` function in [52]. To emphasize quality of fit when p is close to the position of the bumpers, we repeat the samples such that $|p - a_i| \leq 0.1$, $i = 1, 2$, three times. The accuracy achieved by DT is 99.9% on the resulting training dataset. Next, we simulate the system in closed-loop with the DT-based controller from the same initial condition for comparison, obtaining the closed-loop state trajectories shown in Figure 8. While the DT-based controller is still able to track the desired categorical set-point, the CPU time for evaluating the DT-based controller drastically reduces to 52÷67 μs (55 μs on average).

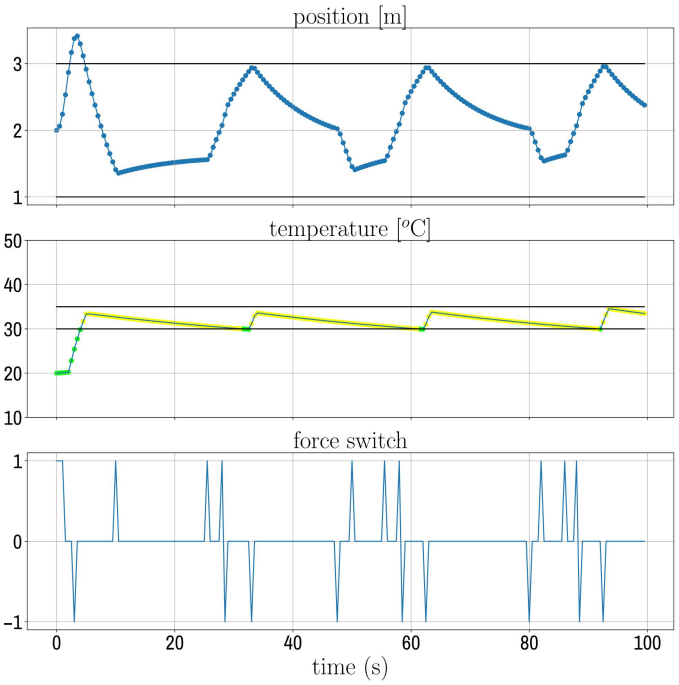


Fig. 8. Toy example: closed-loop results based on a decision-tree approximation of the hybrid MPC controller obtained from PARC with $\boldsymbol{K = 5}$.

VII. Conclusions

The proposed PARC algorithm generalizes linear regression and classification approaches, in particular ridge regression and softmax regression, to piecewise linear form, inheriting their intrinsic robustness with respect to noise on feature and target data. A possible drawback of PARC is its computation time, mainly due to solving a sequence of softmax regression problems. Other regression and classification methods, such as deep neural networks, more complex decision trees, and even random forests may achieve better scores on test data and reduce training time. However, they would return predictors that, compared to the proposed piecewise affine models, are more complicated to optimize globally, therefore complicating the corresponding MPC controller.

The proposed algorithm can be extended in several ways.

[2] `https://coin-or.github.io/pulp/`

For example, ℓ_1-penalties can be introduced in (14) to promote sparsity of a, b. Moreover, in case creating PWL partitions in a reduced set of features is desired for better explanation of the separating hyperplanes, it is enough to zero some columns of vector ω^j in (11a), or alternatively compute the centroids in (12) on the subvector of selected features. The proof of Theorem 1 can be easily extended to cover both modifications. Moreover, basis functions $\phi_i(x)$ can be used instead of x directly, such as canonical piecewise linear functions [16]–[19] to maintain the PWL nature of the predictor, with possibly different basis functions chosen for partitioning the feature space and for fitting targets.

We finally remark that, although we have shown how the PARC algorithm can be used as the main component of a workflow to learn and control hybrid dynamical systems from mixed numeric and categorical data, it is a general supervised learning method that can be applied in several other contexts.

Acknowledgement

The author thanks Faiq Ghanash for a personal communication on the encoding of piecewise affine mappings, that inspired the formulation of Lemma 3.

References

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[2] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[3] H. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964.

[4] D. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[5] E. Brochu, V. Cora, and N. D. Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[6] A. Bemporad, "Global optimization via inverse distance weighting and radial basis functions," *Computational Optimization and Applications*, vol. 77, pp. 571–595, 2020, code available at http://cse.lab.imtlucca.it/~bemporad/glis.

[7] A. Bemporad and D. Piga, "Active preference learning based on radial basis functions," *Machine Learning*, vol. 110, no. 2, pp. 417–448, 2021, code available at http://cse.lab.imtlucca.it/~bemporad/glis.

[8] N. Queipo, R. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Tucker, "Surrogate-based analysis and optimization," *Progress in aerospace sciences*, vol. 41, no. 1, pp. 1–28, 2005.

[9] E. Camacho and C. Bordons, *Model Predictive Control*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer, 1999.

[10] D. Mayne, J. Rawlings, and M. Diehl, *Model Predictive Control: Theory and Design*, 2nd ed. Madison,WI: Nob Hill Publishing, LCC, 2018.

[11] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[12] L. Ljung, *System Identification : Theory for the User*, 2^{nd} ed. Prentice Hall, 1999.

[13] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Systems Magazine*, vol. 39, no. 6, pp. 28–99, 2019.

[14] D. Masti and A. Bemporad, "Learning nonlinear state-space models using autoencoders," *Automatica*, 2020, in press.

[15] L. Breiman, "Hinging hyperplanes for regression, classification, and function approximation," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 999–1013, 1993.

[16] J. Lin and R. Unbehauen, "Canonical piecewise-linear approximations," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 39, no. 8, pp. 697–699, 1992.

[17] L. Chua and A. Deng, "Canonical piecewise-linear representation," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 1, pp. 101–111, 1988.

[18] P. Julián, A. Desages, and B. D'Amico, "Orthonormal high-level canonical PWL functions with applications to model reduction," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 702–712, 2000.

[19] A. Bemporad, A. Oliveri, T. Poggi, and M. Storace, "Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations," *IEEE Trans. Automatic Control*, vol. 56, no. 12, pp. 2883–2897, 2011.

[20] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[21] A. Lodi, "Mixed integer programming computation," in *50 years of integer programming 1958-2008*. Springer, 2010, pp. 619–645.

[22] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, "A clustering technique for the identification of piecewise affine systems," *Automatica*, vol. 39, no. 2, pp. 205–217, Feb. 2003.

[23] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "An algebraic geometric approach to the identification of a class of linear hybrid systems," in *Proc. 42th IEEE Conf. on Decision and Control*, Maui, Hawaii, 2003, pp. 167–172.

[24] J. Roll, A. Bemporad, and L. Ljung, "Identification of piecewise affine systems via mixed-integer programming," *Automatica*, vol. 40, no. 1, pp. 37–50, 2004.

[25] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino, "A bounded-error approach to piecewise affine system identification," *IEEE Trans. Automatic Control*, vol. 50, no. 10, pp. 1567–1580, Oct. 2005.

[26] H. Nakada, K. Takaba, and T. Katayama, "Identification of piecewise affine systems based on statistical clustering technique," *Automatica*, vol. 41, no. 5, pp. 905–913, 2005.

[27] A. Hartmann, J. M. Lemos, R. S. Costa, J. Xavier, and S. Vinga, "Identification of switched ARX models via convex optimization and expectation maximization," *Journal of Process Control*, vol. 28, pp. 9–16, 2015.

[28] F. Lauer, "On the complexity of piecewise affine system identification," *Automatica*, vol. 62, pp. 148–153, 2015.

[29] Y. Yuan, X. Tang, W. Zhou, W. Pan, X. Li, H.-T. Zhang, H. Ding, and J. Goncalves, "Data driven discovery of cyber physical systems," *Nature Communications*, vol. 10, no. 1, pp. 1–9, 2019.

[30] S. Paoletti, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal, "Identification of hybrid systems a tutorial," *European journal of control*, vol. 13, no. 2, pp. 242–260, 2007.

[31] L. Bako, K. Boukharouba, E. Duviella, and S. Lecoeuche, "A recursive identification algorithm for switched linear/affine models," *Nonlinear Analysis: Hybrid Systems*, vol. 5, no. 2, pp. 242–253, 2011.

[32] V. Breschi, D. Piga, and A. Bemporad, "Piecewise affine regression via recursive multiple least squares and multicategory discrimination," *Automatica*, vol. 73, pp. 155–162, Nov. 2016.

[33] K. Bennett and O. Mangasarian, "Multicategory discrimination via linear programming," *Optimization Methods and Software*, vol. 3, pp. 27–39, 1994.

[34] S. Lloyd, "Least square quantization in PCM," *Bell Telephone Laboratories Paper. Also published in IEEE Trans. Inform. Theor., vol. 18, n. 2, pp. 129–137, 1982*, 1957.

[35] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," *IEEE Trans. Automatic Control*, vol. 62, no. 12, pp. 6094–6109, 2017.

[36] M. Schechter, "Polyhedral functions and multiparametric linear programming," *Journal of Optimization Theory and Applications*, vol. 53, no. 2, pp. 269–280, May 1987.

[37] D. Cox, "Some procedures connected with the logistic qualitative response curve," in *Research Papers in Probability and Statistics (Festschrift for J. Neyman)*, F. David, Ed., 1966, pp. 55—71.

[38] H. Thiel, "A multinomial extension of the linear logit model," *International Economic Review*, vol. 10, no. 3, pp. 251–259, 1969.

[39] A. Bemporad, D. Bernardini, and P. Patrinos, "A convex feasibility approach to anytime model predictive control," IMT Institute for Advanced Studies, Lucca, Tech. Rep., Feb. 2015, http://arxiv.org/abs/1502.07974.

[40] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 421–436.

[41] R. Duda and H. Fossum, "Pattern classification by iteratively determined linear and piecewise linear discriminant functions," *IEEE Transactions on Electronic Computers*, no. 2, pp. 220–232, 1966.

[42] J. Spall, "Cyclic seesaw process for optimization and identification," *Journal of Optimization Theory and Applications*, vol. 154, no. 1, pp. 187–208, 2012.

[43] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics*, pp. 1027–1035, 2007.
[44] R. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.
[45] D. O'Leary, "Robust regression computation using iteratively reweighted least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 466–480, 1990.
[46] M. Schmidt, N. L. Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.
[47] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.
[48] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 6, pp. 957–968, 2005.
[49] F. Facchinei, G. Scutari, and S. Sagratella, "Parallel selective algorithms for nonconvex big data optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 7, pp. 1874–1889, 2015.
[50] R. Jyothi and P. Babu, "PIANO: A fast parallel iterative algorithm for multinomial and sparse multinomial logistic regression," *arXiv preprint arXiv:2002.09133*, 2020.
[51] P. Blanchard, D. Higham, and N. Higham, "Accurately computing the log-sum-exp and softmax functions," *MIMS EPrint: 2019.16*, 2019. [Online]. Available: \url{http://eprints.maths.manchester.ac.uk/2765/}
[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[53] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, "PMLB: a large benchmark suite for machine learning evaluation and comparison," *BioData Mining*, vol. 10, no. 1, p. 36, Dec 2017. [Online]. Available: \url{https://epistasislab.github.io/pmlb}
[54] A. Bemporad, "Piecewise linear regression and classification," available on arXiv at https://arxiv.org/abs/2103.06189. Code available at http://cse.lab.imtlucca.it/~bemporad/parc.
[55] Y. Lee and B. Kouvaritakis, "A linear programming approach to constrained robust predictive control," *IEEE Transactions on Automatic Control*, vol. 45, no. 9, pp. 1765–1770, 2000.
[56] A. Bemporad, "Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form," *IEEE Trans. Automatic Control*, vol. 49, no. 5, pp. 832–838, 2004.
[57] ——, *Hybrid Toolbox – User's Guide*, Jan. 2004, http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox.
[58] A. Bemporad, G. Ferrari-Trecate, and M. Morari, "Observability and controllability of piecewise affine and hybrid systems," *IEEE Trans. Automatic Control*, vol. 45, no. 10, pp. 1864–1876, 2000.
[59] G. Ferrari-Trecate, F. Cuzzola, and M. Morari, "Lagrange stability and performance analysis of discrete-time piecewise affine systems with logic states," *International Journal of Control*, vol. 76, no. 16, pp. 1585–1598, 2003.
[60] F. Torrisi and A. Bemporad, "HYSDEL — A tool for generating computational hybrid models," *IEEE Trans. Contr. Systems Technology*, vol. 12, no. 2, pp. 235–249, Mar. 2004.
[61] J. Hooker and M. Osorio, "Mixed logical/linear programming," *Discrete Applied Mathematics*, vol. 96–97, pp. 395–442, 1999.
[62] IBM, Inc., *IBM ILOG CPLEX Optimization Studio 12.10 – User Manual*, 2020.

Alberto Bemporad received his Master's degree in Electrical Engineering in 1993 and his Ph.D. in Control Engineering in 1997 from the University of Florence, Italy. In 1996/97 he was with the Center for Robotics and Automation, Department of Systems Science & Mathematics, Washington University, St. Louis. In 1997-1999 he held a postdoctoral position at the Automatic Control Laboratory, ETH Zurich, Switzerland, where he collaborated as a senior researcher until 2002. In 1999-2009 he was with the Department of Information Engineering of the University of Siena, Italy, becoming an Associate Professor in 2005. In 2010-2011 he was with the Department of Mechanical and Structural Engineering of the University of Trento, Italy. Since 2011 he is Full Professor at the IMT School for Advanced Studies Lucca, Italy, where he served as the Director of the institute in 2012-2015. He spent visiting periods at Stanford University, University of Michigan, and Zhejiang University. In 2011 he cofounded ODYS S.r.l., a company specialized in developing model predictive control systems for industrial production. He has published more than 350 papers in the areas of model predictive control, hybrid systems, optimization, automotive control, and is the co-inventor of 16 patents. He is author or coauthor of various software packages for model predictive control design and implementation, including the Model Predictive Control Toolbox (The Mathworks, Inc.) and the Hybrid Toolbox for MATLAB. He was an Associate Editor of the IEEE Transactions on Automatic Control during 2001-2004 and Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society in 2002-2010. He received the IFAC High-Impact Paper Award for the 2011-14 triennial and the IEEE CSS Transition to Practice Award in 2019. He is an IEEE Fellow since 2010.