Piecewise Affine Regression via Recursive Multiple Least Squares and Multicategory Discrimination

Technical Report TR-IMT-DYSCO-2016-01

Alberto Bemporad, Valentina Breschi, Dario Piga

Foreword

Abstract

In nonlinear regression choosing an adequate model structure is often a challenging problem. While simple models (such as linear functions) may not be able to capture the underlying relationship among the variables, over-parameterized models described by a large set of nonlinear basis functions tend to overfit the training data, leading to poor generalization on unseen data. Piecewise-affine (PWA) models can describe nonlinear and possible discontinuous relationships while maintaining simple local affine regressor-to-output mappings, with extreme flexibility when the polyhedral partitioning of the regressor space is learned from data rather than fixed a priori. In this report, we describe a numerically very efficient two-stage approach for PWA regression based on a combined use of (*i*) recursive multi-model least-squares techniques for clustering and fitting linear functions to data, and (*ii*) piecewise linear multi-category discrimination, either offline (batch) via a Newton-like algorithm for computing a solution of unconstrained optimization problems with objective functions having a piecewise smooth gradient, or online (recursive) via averaged stochastic gradient descent. The resulting PWA regression method is shown very effective in identifying multi-input multi-output hybrid dynamical models and linear parameter-varying models.

The main ideas of the proposed PWA regression algorithm are also descibed in the paper [7], while its application to the identification of hybrid and linear parameter-varying dynamical models is discussed in [12].

1 Introduction

Regression analysis is a supervised learning method which aims at reconstructing the relationship between feature vectors $x \in \mathbb{R}^{n_x}$ and continuous-valued target outputs $y \in \mathbb{R}^{n_y}$ from a set of training data. PieceWise Affine (PWA)

*The authors are with the IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy

functions provide simple yet flexible model structures for nonlinear regression, as they can describe nonlinear and possible discontinuous relationships between the regressor x and the output y. They are defined by partitioning the regressor space into a finite number of polyhedral regions with non-overlapping interiors, and by considering an affine local model on each polyhedron.

The *PWA regression problem* amounts to learning, from a set of training data, both the partition of the regressor space and the parameters defining each affine submodel. PWA regression algorithms were proposed in the last years, mainly for the identification of PWA dynamical systems, see [17, 24] for an overview. Although quite powerful in fitting PWA models to data, most algorithms suffer from computational complexity.

In [25] the authors solve the PWA regression problem via mixed-integer programming. As the number of integer variables increases with the number of training samples, the approach is limited to problems with a small number of observations in which global optimality is sought.

The algorithms proposed in [8, 15, 18, 22] first compute the parameters of the affine local models, then partition of the regressor space. The observations are clustered by assigning each datapoint to a submodel according to a certain criterion, estimating at the same time the parameters of the affine submodels. In a second stage, linear separation techniques are used to compute the polyhedral partition. These algorithms have shown good performance in practice, but can be numerically inefficient.

The *greedy* algorithm of [8] to partition infeasible sets of linear inequalities can be computationally heavy in case of large training sets. The *Expectation Maximization* (EM) algorithm used to numerically implement the statistical clustering method of [22] can become inefficient in case of PWA maps with many parameters. In [18], the submodel parameters are described through probability density functions, which are iteratively updated through particle filtering algorithms; however, an accurate approximation of the probability density functions might require a high number of particles. In [15], the regressor vectors are clustered through a K-means-like algorithm and the submodel parameters are obtained via weighted least-squares. Although [15] is able to handle large training sets both in the clustering and in the parameter estimation phase, poor results might be obtained when the affine local submodels are over-parametrized (i.e., the local models depend on redundant regressors), since the distances in the regressor space (i.e., the only criterion used for clustering) turns out to be corrupted by redundant, thus irrelevant, information.

Another limitation affecting all of the PWA regression algorithms mentioned above is that they can be executed only in a batch mode, and thus they are not suitable for online applications, in which the PWA model must be updated in real-time when new data are acquired. A computationally efficient algorithm for online PWA regression was proposed in [3], where training samples are clustered iteratively and model parameters are updated through recursive least-squares techniques. A main limitation of the approach is that the polyhedral partition of the regressor space is given by the Voronoi diagram of the clusters' centroids, a less flexible structure than general piecewise linear separation maps that may limit regression capabilities.

This report describes a novel approach for approximating vector-valued, and possibly discontinuous, functions in PWA form, trying to overcome the aforementioned limitations of existing algorithms. The proposed algorithm consists of two stages: (**S1**) simultaneous *clustering* of the regressor vectors and *estimation* of the model parameters, performed recursively by processing the training pairs $\{x(k), y(k)\}$ sequentially; (**S2**) *computation of a polyhedral partition* of the regressor space through efficient multi-class linear separation methods, either performed in a batch way via a Newton-like method, or online (recursively) via an averaged stochastic gradient descent algorithm. Overall, the PWA regression algorithm is computationally very effective for offline learning and very suitable for online learning, as will be shown in examples.

The proposed PWA regression approach is employed for the identification of discrete-time multi-input multi-output PWA dynamical systems and *Linear Parameter-Varying* (LPV) systems, in which the main tuning knob is the number of submodels the user must select, trading off quality of fit and model complexity. Overall, we show that the advantages of the proposed PWA regression algorithm over other existing methods is twofold: computationally very effective for offline learning, very suitable for online learning.

The report is organized as follows. The PWA regression problem is formulated in Section 2. First, the general problem of reconstructing a PWA map from a set of experimental data is introduced. The more specific problems of data-driven modeling of discrete-time PWARX models and LPV models are formulated in Section 2.1 and Section 2.2, respectively. Section 3 describes the algorithm used to simultaneously cluster the observed regressors and update the model parameters, and the multi-category discrimination algorithms used to compute the polyhedral partition of the regressor domain. The proposed PWA regression algorithm is tested in Section 4 on five examples. In the first two examples, static nonlinear functions are approximated, based on a set of training samples, with static PWA maps. In the third example, experimental data taken from the UCI machine learning database repository [21] is used to test the performance of the proposed approach on a real case study. The last two simulation examples concern the identification of PWARX models and LPV models.

1.1 Notation

Let $\mathbb{R}^n$ be the set of real vectors of dimension n. Let $I \subset \{1, 2, \dots, \}$ be a finite set of integers and denote by $|I|$ the cardinality of I. Given a vector $a \in \mathbb{R}^n$, let a_i denote the i-th entry of a, a_I the subvector obtained by collecting the entries a_i for all $i \in I$, $\|a\|_2$ the Euclidean norm of a, a_+ a vector whose i-th element is $\max\{a_i, 0\}$. Given two vectors $a, b \in \mathbb{R}^n$, $\max(a, b)$ is the vector whose i-th component is $\max\{a_i, b_i\}$. Given a matrix $A \in \mathbb{R}^{n \times m}$, A' denotes the transpose of A, A_i the i-th row of A, A_I the submatrix of A obtained by collecting the rows A_i for all $i \in I$, $A_{I,J}$ the submatrix of A obtained by collecting the rows and columns of A indexed by $i \in I$ and $j \in J$, respectively. Given two matrices A and B, $A \otimes B$ denotes the Kronecker product between A and B. Let I_n be the identity matrix of size n, and $\mathbf{1}_n$ and $\mathbf{0}_n$ be the n-dimensional column vector of ones and zeros, respectively. The symbol "$\propto$" denotes linear proportionality.

2 Problem statement

Consider a vector-valued PWA function $f : \mathcal{X} \to \mathbb{R}^{n_y}$ defined as

$$f(x) = \begin{cases} A_1[\ 1 \quad x'\ ]' & \text{if } x \in \mathcal{X}_1, \\ \vdots & \\ A_s[\ 1 \quad x'\ ]' & \text{if } x \in \mathcal{X}_s, \end{cases} \tag{1}$$

where $x \in \mathbb{R}^{n_x}$, $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, $s \in \mathbb{N}$ denotes the number of affine local models defining f, $A_i \in \mathbb{R}^{n_y \times (n_x+1)}$ are parameter matrices, and the sets $\mathcal{X}_i$, $i = 1, \ldots, s$ are polyhedra, that form a complete polyhedral partition[1] of the space $\mathcal{X}$. Function f is not assumed to be continuous over the boundaries of the polyhedra $\{\mathcal{X}_i\}_{i=1}^s$; therefore, to avoid that f might take multiple values at the boundaries of $\{\mathcal{X}_i\}_{i=1}^s$, some inequalities can be replaced by strict inequalities in the definition of the sets $\mathcal{X}_i$ to avoid ambiguities when evaluating f on the boundary between neighboring polyhedra.

We address a PWA regression problem, which aims at computing a PWA map f fitting a given set of N input/output pairs $\{x(k), y(k)\}_{k=1}^N$. Computing the PWA map f requires (i) choosing the number of affine submodels s, (ii) computing the parameter matrices $\{A_i\}_{i=1}^s$ that characterize the affine local models of the PWA map f, and (iii) finding the polyhedral partitioning $\{\mathcal{X}_i\}_{i=1}^s$ of the regressor space $\mathcal{X}$ where those local models are defined.

When choosing s, one must take into account the tradeoff between fitting the data and avoiding model complexity and overfit, with consequent poor generalization on unseen data. This is related to one of the most crucial aspects in function learning, known as *bias-variance tradeoff* [29]. In this work, we assume that s is fixed by the user. The value of s can be chosen through cross-validation based procedures, with a possible upper-bound dictated by the maximum tolerable complexity of the estimated model.

PWA regression is quite a general problem and, as described in the following subsections, covers the case of identification of dynamical systems in *PieceWise Affine autoRegressive with eXogenous inputs* (PWARX) form and of *Linear Parameter-Varying* (LPV) systems.

2.1 Identification of PWARX models

The function (1) represents a *Multi-Input Multi-Output* (MIMO) PWARX discrete-time dynamical system if the regressor vector $x(k)$ at the sampling step k is a collection of past input and output observations, i.e.,

$$\begin{aligned} x(k) = [&y'(k-1) \;\; y'(k-2) \;\; \cdots \;\; y'(k-n_a) \\ &u'(k-1) \;\; u'(k-2) \;\; \cdots \;\; u'(k-n_b)]', \end{aligned} \tag{2}$$

where $u(k) \in \mathbb{R}^{n_u}$ and $y(k) \in \mathbb{R}^{n_y}$ denote the observed input and output signals at time k, respectively.

[1]A collection $\{\mathcal{X}_i\}_{i=1}^s$ is a complete partition of the regressor domain $\mathcal{X}$ if $\bigcup_{i=1}^s \mathcal{X}_i = \mathcal{X}$ and $\mathring{\mathcal{X}}_i \cap \mathring{\mathcal{X}}_j = \emptyset$, $\forall i \neq j$, with $\mathring{\mathcal{X}}_i$ denoting the interior of $\mathcal{X}_i$.

2.2 Identification of LPV models

LPV systems can be seen as an extension of *Linear Time-Invariant* (LTI) systems, as the dynamic relation between input and output signals is linear but it can change over time according to a measurable time-varying signal $p(t) \in \mathbb{R}^{n_p}$, the so-called *scheduling variable*. LPV models can accurately describe the dynamic behaviour of a large class of nonlinear and time-varying systems, while preserving a linear relation between the input and output signals. Practical use of LPV models is stimulated by the fact that the theory of LPV control is well established, mainly based on the extension of the results of optimal and robust LTI control theory [2, 23, 27] and its direct use in Model Predictive Control (MPC) [9].

The identification problem of MIMO LPV systems can be formulated as a PWA regression problem by adopting the following MIMO LPV-ARX form

$$\begin{aligned} y(k) = a_0(p(k)) + \sum_{j=1}^{n_a} a_j(p(k))y(k-j) + \\ + \sum_{j=1}^{n_b} a_{j+n_a}(p(k))u(k-j), \end{aligned} \tag{3}$$

where $a_j(p(k))$, $j = 0, \ldots, n_a + n_b$, are PWA functions of $p(k)$ defined as:

$$a_j(p(k)) = \begin{cases} A_1^j(p(k)) & \text{if } p(k) \in \mathcal{P}_1, \\ \vdots & \\ A_s^j(p(k)) & \text{if } p(k) \in \mathcal{P}_s, \end{cases} \tag{4}$$

and $p(k) \in \mathcal{P} \subseteq \mathbb{R}^{n_p}$ is the value of the scheduling variable at time k. By imposing that each entry of matrix $A_i^j(p(k))$ depends affinely on the scheduling variable $p(k)$, the LPV identification problem reduces to reconstructing the PWA mapping of the p-dependent coefficient functions $\{a_j(p(k))\}_{j=0}^{n_a+n_b}$ over the polyhedral partition $\{\mathcal{P}_i\}_{i=1}^{s}$ of the scheduling variable set $\mathcal{P}$. In this case, the N-length sequence of observations $\{x(k), y(k)\}_{k=1}^{N}$ requires $x(k) = [y'(k-1)\ \ldots y'(k-n_a)\ u'(k-1)\ \ldots u'(k-n_b)]' \otimes [1\ p'(k)]'$.

Remark 1 In case the entries of the sub-model matrices $A_i^j(p(k))$ (with $i = 1, \ldots, s$ and $j = 0, \ldots, n_a + n_b$) do not depend on the scheduling variable $p(k)$ (i.e., the p-dependent functions $a_j(p(k))$ are approximated by piecewise constant functions), the LPV model (3)-(4) becomes a switched linear model, where each subsystem is LTI and the switching behaviour depends on the scheduling signal $p(k)$. ■

Remark 2 Since the polyhedral partition $\{\mathcal{P}_i\}_{i=1}^{s}$ is not fixed a priori, the underlying dependencies of the functions $a_j(p(k))$ on the scheduling variable $p(k)$ are directly reconstructed from data. This represents one of the main advantages w.r.t. widely used parametric LPV identification approaches (see, e.g., [4, 28]), which in turn require to parameterize $a_j(p(k))$ as a linear combination of some basis functions specified a priori (e.g., polynomial or trigonometric functions). Indeed, the choice of the number and type of basis functions is a critical issue in the identification procedure. In fact, an inaccurate selection of the set of basis functions could lead to a structural bias. Therefore, in order to capture the underlying dependence of the coefficient functions $a_j(p(k))$ on the scheduling variable $p(k)$, a large set of basis functions is often used. As a consequence, besides numerical problems in the implementation of the identification schemes, overfitting may occur and the estimated model may show poor generalization to unseen data. ■

3 PWA regression algorithm

As mentioned in Section 1, the general PWA regression problem will be tackled in two stages: S1 (iterative clustering and parameter estimation) and S2 (polyhedral partition of the regressor space).

3.1 Recursive clustering and parameter estimation

Stage S1 is carried out as described in Algorithm 1. The algorithm is an extension to the case of multiple linear regressions and clustering of the (computationally very efficient) approach proposed in [1] for solving recursive least squares problems using inverse QR decomposition. Algorithm 1 updates the clusters and the model parameters iteratively, and thus it is also suitable for online applications, when data are acquired in real-time.

The algorithm requires an initial guess for the parameter matrices A_i and cluster centroids c_i, $i = 1, \ldots, s$. Because of the greedy nature of Algorithm 1, the final estimate depends on the chosen initial conditions, and no fit criterion to minimize $\{\|y(k) - f(x(k))\|\}_{k=1}^{N}$ is optimized. Zero matrices A_i, randomly chosen centroids c_i, and identity covariance matrices R_i are a possible initialization. In alternative, if Algorithm 1 can be executed in a batch mode, one can initialize the parameter matrices $A_1, \ldots, A_s$ all equal to the best linear model

$$A_i \equiv \arg\min_{A} \sum_{k=1}^{N} \|y(k) - A \left[\begin{smallmatrix} 1 \\ x(k) \end{smallmatrix}\right] \|_2^2, \ \forall i = 1, \ldots, s \tag{5}$$

that fits all data, classify the regressors $\{x(k)\}_{k=1}^{N}$ through k-means clustering, compute the cluster centroids $c_i = \frac{1}{|\mathcal{C}_i|} \sum_{x(k) \in \mathcal{C}_i} x(k)$ and the cluster covariance matrices $R_i = \frac{1}{|\mathcal{C}_i| - 1} \sum_{x(k) \in \mathcal{C}_i} [x(k) - c_i] [x(k) - c_i]'$. When working in a batch mode, estimation quality may be improved by repeating Algorithm 1 iteratively, using its output as initial condition for its following execution. It is worth remarking that a prior knowledge of the noise covariance matrix Λ_e is barely available in practice. A possible choice for Λ_e can be, for instance, $\Lambda_e = I_{n_y}$ (i.e., the regression errors are not weighted in eq. (A1.1)). Alternatively, if Algorithm 1 is executed in a batch mode and iteratively repeated by using the output as initial condition for the next execution, an estimate $\hat{\Lambda}_e$ of Λ_e can be computed at the end of each execution as the sample covariance matrix

$$\hat{\Lambda}_e = \frac{1}{N} \sum_{i=1}^{s} \sum_{\substack{k=1 \\ x(k) \in \mathcal{C}_i}}^{N} \left(y(k) - A_i \left[\begin{smallmatrix} 1 \\ x(k) \end{smallmatrix}\right]\right) \left(y(k) - A_i \left[\begin{smallmatrix} 1 \\ x(k) \end{smallmatrix}\right]\right)'.$$

Algorithm 1 Recursive clustering and parameter estimation algorithm

Input: Observations $\{x(k), y(k)\}_{k=1}^{N}$, desired number s of affine submodels, noise covariance matrix Λ_e, forgetting factor κ, $0 < \kappa \leq 1$, inverse matrix init parameter δ, $\delta \gg 1$; initial condition for matrices A_i, cluster centroids c_i, and covariance matrices R_i, $i = 1, \ldots, s$.

```
1. let $\mathcal{C}_i \leftarrow \emptyset$, $i = 1, \ldots, s$;
2. let $T^{i,j}(0) \leftarrow \delta I_{n_x+1}$, $j = 1, \ldots, n_y$, $i = 1, \ldots, s$;
3. for $k = 1, \ldots, N$ do
  3.1. let $e_i(k) \leftarrow y(k) - A_i \left[\begin{smallmatrix} 1 \\ x(k) \end{smallmatrix}\right]$, $i = 1, \ldots, s$;
  3.2. let
       $i(k) \leftarrow \arg\min_{i=1,\ldots,s} (x(k)-c_i)' R_i^{-1} (x(k)-c_i) + e_i(k)' \Lambda_e^{-1} e_i(k)$;     (A1.1)
  3.3. let $\mathcal{C}_{i(k)} \leftarrow \mathcal{C}_{i(k)} \cup \{x(k)\}$;
  3.4. for $j = 1, \ldots, n_y$ do
    3.4.1. let $u \leftarrow \mathbf{0_{n_x+1}}$, $b \leftarrow 1$;
    3.4.2. for $\ell = 1, \ldots, n_x + 1$ do
      3.3.4.1. $a \leftarrow \frac{1}{\sqrt{\kappa}} \sum_{h=1}^{\ell} [T^{i(k),j}]_{\ell,h} x_h(k)$;
      3.3.4.2. $b_1 \leftarrow b$; $b \leftarrow \sqrt{b^2 + a^2}$;
      3.3.4.3. $\sigma \leftarrow \frac{a}{b}$, $\rho \leftarrow \frac{b_1}{b}$;
      3.3.4.4. for $t = 1, \ldots, i$ do
               $d \leftarrow [T^{i(k),j}]_{\ell,t}$; $[T^{i(k),j}]_{\ell,t} \leftarrow \frac{1}{\sqrt{\kappa}} \rho d - \sigma u_t$;
               $u_t \leftarrow \rho u_t + \frac{1}{\sqrt{\kappa}} \sigma d$;
      3.3.4.5. end for;
    3.4.3. end for;
    3.4.4. update $[A_{i(k)}]_{j,:} \leftarrow [A_{i(k)}]_{j,:} + \frac{e_{i(k)}}{b} u'$;
  3.5. end for;
  3.6. let $\delta c_{i(k)} \leftarrow \frac{1}{|\mathcal{C}_{i(k)}|} (x(k) - c_{i(k)})$;
  3.7. update the centroid $c_{i(k)}$ of cluster $\mathcal{C}_{i(k)}$
       $c_{i(k)} \leftarrow c_{i(k)} + \delta c_{i(k)}$;
  3.8. update the cluster covariance matrix $R_{i(k)}$ for cluster $\mathcal{C}_{i(k)}$
       $R_{i(k)} \leftarrow \frac{|\mathcal{C}_{i(k)}| - 2}{|\mathcal{C}_{i(k)}| - 1} R_{i(k)} + \delta c_{i(k)} \delta c'_{i(k)} +$
       $\frac{1}{|\mathcal{C}_{i(k)}| - 1} \left[x(k) - c_{i(k)}\right] \left[x(k) - c_{i(k)}\right]'$;
4. end for;
5. end.
```

Output: Estimated matrices $\{A_i\}_{i=1}^{s}$, centroids $\{c_i\}_{i=1}^{s}$, clusters $\{\mathcal{C}_i\}_{i=1}^{s}$, covariance matrices $\{R_i\}_{i=1}^{s}$.

Step 2 initializes the inverse matrices $T^{i,j}$ needed by the recursive least squares updates at a value δI_{n_x+1}, where δ is a large number, for all output components $j = 1, \dots, n_y$ and for all local linear models $i = 1, \dots, s$.

After computing the estimation error $e_i(k)$ for all models i at Step 3.1, Step 3.2 picks up the "best" submodel $i(k)$ to which the current sample $x(k)$ must be associated with, based on a tradeoff between reducing the prediction error $e_i(k)$ and penalizing the distance (weighted by matrix R_i^{-1}) between $x(k)$ and the corresponding centroid c_i. This is motivated by the stochastic interpretation described in Section 3.1.1. The clustering rule in eq. (A1.1) is similar to the clustering criterion used in [3] for online PWA regression. However, [3] does not provide guidance to properly weight the prediction error and the distance from the cluster centroids. Instead, we motivate the use of the chosen weighting parameters in Section 3.1.1.

Steps 3.4.1–3.4.4 are derived from the inverse QR factorization algorithm of [1] and recursively update each row ℓ of matrix $A_{i(k)}$ only for the selected submodel $i(k)$, for all $\ell = 1, \dots, n_y$. Step 3.6 updates recursively the corresponding centroid $c_{i(k)}$ and the corresponding cluster covariance matrix $R_{i(k)}$. Note that the remaining clusters' centroids and covariance matrices are not updated.

3.1.1 Cluster selection: a stochastic interpretation

In eq. (A1.1) each vector $x(k)$ is assigned to a cluster $\mathcal{C}_{i(k)}$ by trading off between minimizing the (weighted) regression error and the (weighted) distance between $x(k)$ and the clusters' centroids. This criterion is justified by the following stochastic interpretation. Assume that the conditional *probability density function* f_x of $x(k)$ given that $x(k)$ belongs to the cluster $\mathcal{C}_i$ is a Gaussian function centered at the centroid c_i with covariance matrix R_i, i.e.,

$$f_x(x(k)|x(k)\in\mathcal{C}_i) \propto \exp\left\{-\frac{1}{2}\,(x(k)-c_i)'R_i^{-1}\,(x(k)-c_i)\right\}.$$

Further, suppose that the residual $e(k) = y(k) - A_i \left[\begin{smallmatrix}1\\x(k)\end{smallmatrix}\right]$ given that $x(k)$ belongs to the cluster $\mathcal{C}_i$ follows a Gaussian distribution with zero mean and covariance matrix Λ_e. Thus, the conditional *probability density function* f_y of the observed output $y(k)$ given the regressor $x(k)$ and the information that $x(k)$ belongs to the cluster $\mathcal{C}_i$ is:

$$\begin{aligned}&f_y(y(k)|x(k), x(k) \in \mathcal{C}_i) = f_y\left(y(k) - A_i \left[\begin{smallmatrix}1\\x(k)\end{smallmatrix}\right]\right) \propto \\ &\exp\left\{-\frac{1}{2}\left(y(k) - A_i\left[\begin{smallmatrix}1\\x(k)\end{smallmatrix}\right]\right)' \Lambda_e^{-1}\left(y(k) - A_i\left[\begin{smallmatrix}1\\x(k)\end{smallmatrix}\right]\right)\right\}.\end{aligned}$$

The criterion in eq. (A1.1) thus maximizes over $i = 1, \dots, s$ the conditional posterior probability $f_{xy}(x(k), y(k)|x(k) \in \mathcal{C}_i)$, which is given by:

$$\begin{aligned}&f_{xy}(x(k), y(k)|x(k) \in \mathcal{C}_i)) = \\ &= f_y(y(k)|x(k), x(k) \in \mathcal{C}_i) f_x(x(k)|x(k) \in \mathcal{C}_i) \propto \\ &\exp\left\{-\frac{1}{2}\left(y(k) - A_i\left[\begin{smallmatrix}1\\x(k)\end{smallmatrix}\right]\right)' \Lambda_e^{-1}\left(y(k) - A_i\left[\begin{smallmatrix}1\\x(k)\end{smallmatrix}\right]\right) + \right. \\ &\qquad \left. -\frac{1}{2}\,(x(k) - c_i)'\, R_i^{-1}\,(x(k) - c_i)\right\}.\end{aligned}$$

3.2 Partitioning the regressor space

We propose here a variation of the multicategory discrimination technique of [10] to separate the clusters $\{\mathcal{C}_i\}_{i=1}^s$ that partitions the regressor space in a much more computationally efficient way, especially when dealing with a large number N of data points.

For $i = 1, \dots, s$, let M_i be a $m_i \times n_x$ dimensional matrix (with m_i denoting the cardinality of cluster $\mathcal{C}_i$) obtained by stacking the regressors $x(k)'$ belonging to $\mathcal{C}_i$ in its rows. The linear multicategory discrimination problem aims at computing a piecewise affine separator function $\phi : \mathbb{R}^{n_x} \to \mathbb{R}$ discriminating between the clusters $\mathcal{C}_1, \dots, \mathcal{C}_s$. The piecewise affine separator ϕ is defined as the maximum of s affine functions $\{\phi_i(x)\}_{i=1}^s$, i.e.,

$$\phi(x) = \max_{i=1,\dots,s} \phi_i(x), \tag{6}$$

and, based on the definition of ϕ, each polyhedron $\mathcal{X}_i$ turns out to be described by:

$$\mathcal{X}_i = \{x \in \mathbb{R}^{n_x} : \phi_i(x) = \phi(x)\}. \tag{7}$$

The affine functions $\phi_i(x)$ are described by the parameters $\omega^i \in \mathbb{R}^{n_x}$ and $\gamma^i \in \mathbb{R}$, namely:

$$\phi_i(x) = [x' \quad -1] \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix}. \tag{8}$$

In case of piecewise linearly separable clusters, the affine functions $\phi_i(x)$ satisfy the inequality constraints

$$\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} > \begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix}, \quad i, j = 1, \dots, s, \ i \neq j,$$

or, equivalently,

$$\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} \geq \begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + \mathbf{1}_{m_i}, \quad i, j = 1, \dots, s, \ i \neq j, \tag{9}$$

where the constant vector $\mathbf{1}_{m_i}$ on the right side of eq. (9) is used only for normalization purposes.

The piecewise-affine separator ϕ thus satisfies the conditions:

$$\begin{cases} \phi(x) = & [x' \quad -1] \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix}, \ \forall x \in \mathcal{C}_i, \quad i = 1, \dots, s \\ \phi(x) \geq & [x' \quad -1] \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + 1, \ \forall x \in \mathcal{C}_i, \ i \neq j \end{cases} \tag{10}$$

Remark 3 According to the definition of ϕ (eq. (6)) and ϕ_i (eq. (8)), and the conditions in (10), the polyhedra $\{\mathcal{X}_i\}_{i=1}^s$ are defined as

$$\mathcal{X}_i = \left\{ x \in \mathbb{R}^{n_x} : [x' \quad -1] \begin{bmatrix} \omega^i - \omega^j \\ \gamma^i - \gamma^j \end{bmatrix} \geq 1, \ j = 1, \dots, s, \ j \neq i \right\}.$$

■

Rather than solving a linear program as in [10], we determine $\{\omega^i, \gamma^i\}_{i=1}^s$ by solving the convex unconstrained optimization problem

$$\min_{\{\omega^i,\gamma^i\}_{i=1}^s} \frac{\lambda}{2}\sum_{i=1}^s \left(\|\omega^i\|_2^2 + (\gamma^i)^2\right) + \tag{11}$$
$$\sum_{i=1}^s \sum_{\substack{j=1\\ j\neq i}}^s \frac{1}{m_i}\left\|\left(\begin{bmatrix} M_i & -\mathbf{1}_{m_i}\end{bmatrix}\begin{bmatrix}\omega^j-\omega^i\\ \gamma^j-\gamma^i\end{bmatrix} + \mathbf{1}_{m_i}\right)_+\right\|_2^2,$$

where $\frac{\lambda}{2}\sum_{i=1}^s \left(\|\omega^i\|_2^2 + (\gamma^i)^2\right)$, with $\lambda > 0$, is an ℓ_2-regularization term introduced to better conditioning problem (11) and to guarantee that (11) has a unique solution (no regularization in case $\lambda = 0$).

Problem (11) generates a piecewise-affine function that minimizes the (averaged) squared 2-norm of the violation of the inequalities (9). The problem is solved by using a regularized piecewise-smooth Newton method with Armijo's line search similar to the one proposed in [6] for functions $g : \mathbb{R}^{n_\xi} \to \mathbb{R}$ of the form

$$g(\xi) = \frac{\lambda}{2}\|\xi\|_2^2 + \sum_{j=1}^{n_g} \|g_j(\xi)_+\|_2^2, \tag{12}$$

where $g_j : \mathbb{R}^{n_\xi} \to \mathbb{R}$ are convex and twice continuously differentiable functions. In particular, we exploit the linearity of functions g_j's. In fact, for the special case of solving Problem (11), the optimization vector is $\xi = [(\omega^1)' \ \dots \ (\omega^s)' \ \gamma^1 \ \dots \ \gamma^s]' \in \mathbb{R}^{n_\xi}$, $n_\xi = s(n_x + 1)$, and g_j's are affine functions:

$$g_j(\xi) = a_j'\xi - b_j, \ j = 1, \dots, n_g, \tag{13}$$

where $n_g = N(s-1)$ and $a_j \in \mathbb{R}^{n_\xi}$, $b_j \in \mathbb{R}$ are easily obtained from (11) as a function of matrices $\{M_i\}_{i=1}^s$ and coefficients $\{m_i\}_{i=1}^s$. By letting

$$\mathcal{A} = [a_1 \ \dots \ a_{ng}]', \ \mathcal{B} = [b_1 \ \dots b_{n_g}]', \tag{14}$$

given a vector $\xi \in \mathbb{R}^{n_\xi}$, let $I(\xi) = \{i \in \{1, \dots, n_g\} : \mathcal{A}_i\xi - \mathcal{B}_i > 0\}$. Then,

$$g(\xi) = \frac{\lambda}{2}\xi'\xi + \sum_{i\in I(\xi)} (\mathcal{A}_i\xi - \mathcal{B}_i)^2 \tag{15a}$$
$$\nabla g(\xi) = \lambda\xi + \mathcal{A}_{I(\xi)}'(\mathcal{A}_{I(\xi)}\xi - \mathcal{B}_{I(\xi)}) \tag{15b}$$
$$\nabla^2 g(\xi) = \lambda I + \mathcal{A}_{I(\xi)}'\mathcal{A}_{I(\xi)} = \lambda I + \sum_{i\in I(\xi)} \mathcal{A}_i'\mathcal{A}_i \tag{15c}$$

are, respectively, the function to minimize, its gradient, and its generalized Hessian at ξ.

The proposed approach to solve (11) is summarized in Algorithm 2. The algorithm uses the solution d of the linear system

$$(\nabla^2 g(\xi) + \delta(\xi)I)d = -\nabla g(\xi) \tag{16}$$

at the current ξ as a search direction, where $\delta(\xi) = \zeta\|\nabla g(\xi)\|$ and $\zeta \in (0, 1)$. Due to the special structure of $\nabla^2 g$ in (15c), the linear system (16) is solved at

Steps 5.1–5.2 as the least squares problem

$$\min_d \frac{1}{2}\left\| \begin{bmatrix} \mathcal{A}_{I(\xi)} \\ \sqrt{\lambda+\delta(\xi)} I_{n_\xi} \end{bmatrix} d + \begin{bmatrix} \mathcal{A}_{I(\xi)}\xi - \mathcal{B}_{I(\xi)} \\ \frac{\lambda}{\sqrt{\lambda+\delta(\xi)}}\xi \end{bmatrix} \right\|_2^2 \tag{17}$$

using the QR factorization of $\begin{bmatrix} \mathcal{A}_{I(\xi)} \\ \sqrt{\lambda+\delta(\xi)} I_{n_\xi} \end{bmatrix}$.

Note that since $\nabla g(\xi) > 0$ during iterations, $\delta(\xi)$ is also positive, and therefore R is full column rank, so that the upper-triangular linear system in Step 5.2 is always solvable.

A good initial guess for $\xi \in \mathbb{R}^n$ can be obtained by running Algorithm 2 first on decimated clusters, then use the result as the new initial condition in Algorithm 2 for the full problem with N regressors.

Numerical experiments have shown that allowing a varying $\zeta = \zeta_0 \frac{\min\{1, \|\nabla g\|\}}{\|\nabla g\|}$, where $0 < \zeta_0 \ll 1$, reduces the number of iterations and prevents excessive regularization in (16) when $\|\nabla g\|$ is large. Moreover, while setting $\lambda > 0$ complicates the number of operations required by the algorithm at each iteration (in particular to compute the solution of eq. (A2.1)) and bias the solution, it leads to a smaller number k of iterations, and overall to a reduced computation time.

3.3 Recursive multicategory discrimination via online convex programming

As an alternative to Algorithm 2, or in addition to it for refining the partition ϕ on line based on streaming data, we introduce a recursive approach to solve problem (11) based on techniques of online convex programming.

Let us treat the data-points $x \in \mathbb{R}^{n_x}$ as random vectors and assume that an oracle function $i : \mathbb{R}^{n_x} :\to \{1, \ldots, s\}$ exists that to any $x \in \mathbb{R}^{n_x}$ assigns the corresponding mode $i(x) \in \{1, \ldots, s\}$. Function i implicitly defines clusters in the data-point space $\mathbb{R}^{n_x}$. Let us also assume that the following values

$$\pi_i = \text{Prob}[i(x) = i] = \int_{\mathbb{R}^{n_x}} \delta(i, i(x)) p(x) dx$$

are known for all $i = 1, \ldots, s$, where $\delta(i,j) = 1$ if $i = j$ or zero otherwise, $i, j \in \{1, \ldots, s\}$. Each value π_i represents the relative "volume" of cluster i, where clearly

$$\sum_{i=1}^s \pi_i = \int_{\mathbb{R}^{n_x}} \sum_{i=1}^s \delta(i, i(x)) p(x) dx = \int_{\mathbb{R}^{n_x}} p(x) dx = 1.$$

Problem (11)–(12) can be generalized to the following unconstrained convex stochastic optimization problem

$$\xi^* = \min_\xi E_{x \in \mathbb{R}^{n_x}} [\ell(x, \xi)] + \frac{\lambda}{2}\|\xi\|_2^2 \tag{18}$$

$$\ell(x,\xi) = \sum_{\substack{j=1 \\ j \neq i(x)}}^{s} \frac{1}{\pi_{i(x)}} \left(x'(\omega^j - \omega^{i(x)}) - \gamma^j + \gamma^{i(x)} + 1 \right)_+^2$$

Algorithm 2 Piecewise-smooth Newton method for solving the multicategory discrimination problem (11)

Input: Regressors $\{x(k)\}_{k=1}^{N}$, clusters $\mathcal{C}_i$, $i=1,\ldots,s$; scalars $\sigma\in(0,1/2)$, $\zeta\in(0,1)$; ℓ_2-regularization hyper-parameter $\lambda\geq 0$; initial guess $\xi\in\mathbb{R}^n$; maximum number K of iterations; tolerances $g_{\rm tol}>0$ and $\delta_{\rm tol}>0$.

```
1. **Initialize** matrices $M_i\in\mathbb{R}^{m_i\times n_x}$, whose rows are the transposed regressors $x(k)\in\mathcal{C}_i$, $i=1,\ldots,s$; $n_\xi\leftarrow s(n_x+1)$, $n_g\leftarrow N(s-1)$; define $\mathcal{A}$, $\mathcal{B}$ as in (13)–(14), $j=1,\ldots,n_g$;
2. $k\leftarrow 0$;
3. $c\leftarrow\mathcal{A}\xi-\mathcal{B}$; $I\leftarrow\{i\in\{1,\ldots,n_g\}:\ c_i\geq 0\}$;
4. $g\leftarrow c_I'c_I+\frac{\lambda}{2}\xi'\xi$; $\nabla g\leftarrow\mathcal{A}_I'c_I+\lambda\xi$; $\delta\leftarrow\zeta\|\nabla g\|$;
5. **while** $g>g_{\rm tol}$ **and** $\delta>\delta_{\rm tol}$ **and** $k<K$ **do**
   5.1. $(Q,R)\leftarrow$ QR factorization of $\begin{bmatrix}\mathcal{A}_I\\ \sqrt{\lambda+\delta}I_{n_\xi}\end{bmatrix}$;
   5.2. **solve** the upper-triangular linear system
        $$\begin{aligned}R_{\{1,\ldots,n_\xi\}}d=&-(Q_{\{1,\ldots,|I|\},\{1,\ldots,n_\xi\}})'c_I\\&-\frac{\lambda}{\lambda+\delta}(Q_{\{|I|+1,\ldots,|I|+n_\xi\},\{1,\ldots,n_\xi\}})'\xi;\end{aligned}\tag{A2.1}$$
   5.3. $\alpha\leftarrow 1$; $q\leftarrow\mathcal{A}d$; $\xi_\alpha\leftarrow\xi+d$;
   5.4. $I_\alpha\leftarrow\{i\in\{1,\ldots,n_g\}:\ c+q\geq 0\}$;
   5.5. $g_\alpha\leftarrow(c_{I_\alpha}+q_{I_\alpha})'(c_{I_\alpha}+q_{I_\alpha})+\frac{\lambda}{2}\xi_\alpha'\xi_\alpha$;
   5.6. **while** $g_\alpha>g+\alpha\sigma\nabla g'd$ **do**
      5.6.1. $\alpha\leftarrow\frac{1}{2}\alpha$; $\xi_\alpha\leftarrow\xi+\alpha d$
      5.6.2. $c^\alpha\leftarrow c+\alpha q$;
      5.6.3. $I_\alpha\leftarrow\{i\in\{1,\ldots,n_g\}:\ c_i^\alpha\geq 0\}$;
      5.6.4. $g_\alpha\leftarrow(c_{I_\alpha}^\alpha)'c_{I_\alpha}^\alpha+\frac{\lambda}{2}\xi_\alpha'\xi_\alpha$;
   5.7. **end while**;
   5.8. $\xi\leftarrow\xi_\alpha$; $g\leftarrow g_\alpha$; $I\leftarrow I_\alpha$; $c\leftarrow c_\alpha$;
   5.9. $\nabla g\leftarrow\mathcal{A}_{I_\alpha}'c_{I_\alpha}^\alpha+\lambda\xi$; $\delta\leftarrow\zeta\|\nabla g\|$;
   5.10. $k\leftarrow k+1$;
6. **retrieve** $\omega^i,\gamma^i$, $i=1,\ldots,s$, from the solution $\xi$;
7. **end**.
```

Output: Coefficients ω^i,γ^i, $i=1,\ldots,s$ defining the piecewise affine separator ϕ in (6)–(8).

with $E_x[\cdot]$ denoting the expected value w.r.t. x. The solution of problem (18) provides the piecewise affine multicategory discrimination function (6)–(8). This aims at violating the least, on average over x, the condition in (9) for $i = i(x)$. We assume that the ℓ_2-regularization hyper-parameter λ is such that $\lambda > 0$, so that the objective function in (18) is strongly convex[2].

When learning the discrimination function ϕ online, the data-points x_k are acquired in real-time and one would like to update ϕ recursively, without the need of storing all past data-points $x_0, \ldots, x_{k-1}$. We achieve this by solving problem (18) by online convex optimization, and in particular the averaged stochastic gradient descent method of [26] as proposed in [11] (cf. also [30]), whose application to the linear multicategory discrimination problem (18) is described in Algorithm 3.

Algorithm 3 Averaged stochastic gradient descent algorithm for solving the linear multicategory discrimination problem (18)

Input: Regressor flow $x(0), x(1), \ldots$; cluster assignment function $i : \mathbb{R}^{n_x} \to \{1, \ldots, s\}$; ℓ_2-regularization hyper-parameter $\lambda > 0$; scalar $\nu_0 \geq 0$; initial guess $\xi \in \mathbb{R}^n$;

```
1. for $k = 0, 1, \ldots$ do:
   1.1. compute the gradient $\nabla_\xi \ell(\xi_k, x_k)$ as follows:
        1.1.1. $I_k \leftarrow \{j \in \{1, \ldots, s\}, j \neq i(x_k) : \ x_k'(\omega_k^j - \omega_k^{i(x_k)}) - \gamma_k^j + \gamma_k^{i(x_k)} \geq -1\}$;
        1.1.2. set
$$\frac{\partial \ell(\xi_k, x_k)}{\partial \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix}} \leftarrow \lambda \begin{bmatrix} \omega_k^j \\ \gamma_k^j \end{bmatrix} + \frac{1}{\pi_{i(x_k)}} \times \begin{cases} \sum_{j \in I_k} \left( x_k'(\omega_k^j - \omega_k^{i(x_k)}) - \gamma_k^j + \gamma_k^{i(x_k)} + 1 \right) \begin{bmatrix} -x_k \\ 1 \end{bmatrix} \\ \qquad \text{if } j = i(x_k) \\ \left( x_k'(\omega_k^j - \omega_k^{i(x_k)}) - \gamma_k^j + \gamma_k^{i(x_k)} + 1 \right) \begin{bmatrix} x_k \\ -1 \end{bmatrix} \\ \qquad \text{if } j \neq i(x_k),\ j \in I_k \\ 0 \text{ otherwise.} \end{cases}$$
   1.2. compute
$$\nu_k \leftarrow \nu_0(1 + \nu_0 \lambda k)^{-\frac{3}{4}}; \quad \mu_k \leftarrow 1/\max\{1, k - n_x, k - n_\xi\};$$
$$\xi_{k+1} \leftarrow \xi_k - \nu_k \nabla_\xi \ell(\xi_k, x_k); \quad \bar{\xi}_{k+1} \leftarrow \bar{\xi}_k + \mu_k(\xi_{k+1} - \bar{\xi}_k);$$
   1.3. retrieve $\omega_k^i, \gamma_k^i$, $i = 1, \ldots, s$, from $\bar{\xi}_k$;
2. end.
```

Output: Coefficients $\{\omega_k^i, \gamma_k^i\}_{i=1}^s$, defining the separator ϕ in (6)–(8) at each step $k = 0, 1, \ldots$.

[2]A differentiable function f is strongly convex if, for all points x,y in its domain, $\exists\ m > 0$ such that $f(y) \geq f(x) + \nabla f(x)^\top (y - x) + m\|x - y\|_2^2$.

The initial estimate ξ_0 can be either zero (or any other value), or the result of the execution of the batch Algorithm 2 on a subset of data preprocessed offline. The coefficients π_i can also be estimated from offline data, namely $\pi_i = \frac{m_i}{N}$, and also possibly updated while Algorithm 3 is running. Numerical experiments have shown however that constant and uniform coefficients $\pi = \frac{1}{s}$ work equally well.

4 Examples

We consider five examples, one of which using experimental data, to test the effectiveness of the proposed PWA regression algorithm and to compare its performance with other existing regression approaches.

All computations are carried out on an i7 2.40-GHz Intel core processor with 4 GB of RAM running MATLAB R2014b. In the considered simulation examples, the output used in the training phase is corrupted by an additive zero-mean white noise $e_{\rm o}$ with Gaussian distribution. The effect of measurement noise on the output signal is quantified through the Signal-to-Noise Ratio (SNR), that is defined for the i-th output channel as

$$\text{SNR}_i = 10 \log \frac{\sum_{k=1}^{N} (y_i(k) - e_{{\rm o},i}(k))^2}{\sum_{k=1}^{N} e_{{\rm o},i}^2(k)}, \tag{19}$$

with $e_{{\rm o},i}(k)$ denoting the i-th component of $e_{\rm o}(k)$.

In validating the obtained models on a data sequence not used for training, we will denote by $y_{\rm o}$ and $\hat{y}$ the vector stacking the measured and estimated outputs, respectively, by $\bar{y}_{\rm o}$ the vector staking the sample mean of the measured output, and use the *Best Fit Rate* (BFR) indicator

$$\text{BFR} = \max\left\{1 - \frac{\|y_{\rm o} - \hat{y}\|_2}{\|y_{\rm o} - \bar{y}_{\rm o}\|_2}, 0\right\} \cdot 100\ \% \tag{20}$$

to assess model quality.

4.1 Example 1: Identification of a mono-dimensional PWA map

Let the data be generated by the following (unknown) function

$$f_{\rm o}(x) = \begin{cases} \begin{bmatrix} 1 & 0.2 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} & \text{if } x \in (-\infty, -1), \\ \begin{bmatrix} -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} & \text{if } x \in [-1, 2), \\ \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} & \text{if } x \in (2, 4], \\ \begin{bmatrix} 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} & \text{if } x \in [4, 6), \\ \begin{bmatrix} 2 & -10 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} & \text{if } x \in [6, \infty), \end{cases} \tag{21}$$

which is proposed as an example for PWA regression in the Hybrid Identification Toolbox [14]. The map (23) is characterized by $s_{\rm o} = 5$ affine submodels. The regressor $x(k) \in \mathbb{R}$ is a white noise sequence with uniform distribution in the interval $[-4,\ 8]$ and length $N = 1000$. The output of the function $f_{\rm o}$ is corrupted by an additive zero-mean white noise $e_{\rm o}(k) \in \mathbb{R}$, with Gaussian distribution and variance $\Lambda_e = 0.01$ (i.e., $y(k) = f_{\rm o}(x(k)) + e_{\rm o}(k)$). This corresponds to a *Signal-to-Noise Ratio* (SNR) of 27 dB.

Table 1: Example 1: true parameters A_i, $i = 1, \ldots, 5$ vs parameters estimated with Algorithms 1-2 and the Algorithm of [15].

	A_1		A_2		A_3		A_4		A_5	
true	1	0.2	-1	2	1	-1	0	2	2	-10
Algorithms 1–2	1.0102	0.2184	-0.9865	1.9983	0.9983	-1.0054	-0.0007	2.0045	1.9954	-9.9668
Algorithm [15]	1.0308	0.2820	-0.9522	1.9794	1.1993	-1.7159	-0.0179	2.0995	2.0000	-10.0014

We run Algorithm 1 with $s = s_o = 5$, with initial guess A_i as in (5) and $\Lambda_e = 1$. No forgetting factor is considered (i.e., $\kappa = 1$) and δ is set equal to 10^3. The initial guess for the cluster centroids and covariance matrices $\{R_i\}_{i=1}^s$ are computed by running an instance of Algorithm 1 without the first term in eq. (A1.1). Algorithm 1 is then run again 5 times, with the full criterion (A1.1), by initializing A_i, c_i and R_i with the output of the previous run. The clusters generated by Algorithm 1 are then separated by solving problem (11) via the Piecewise-smooth Newton method described in Algorithm 2, with parameters $K = 300$, $\sigma = 0.4$, $\lambda = 10^{-5}$, $\zeta = 10^{-4}$, $g_{\text{tol}} = \delta_{\text{tol}} = 10^{-6}$, and $\xi = 0$ as initial guess.

The quality of the estimated PWA model is assessed w.r.t. a validation dataset of noiseless $N_V = 200$ samples. The obtained BFR is 97.05 %. The estimated polyhedral partition of the regressor space is such that only 1 out of 200 data samples are misclassified. The total CPU time for solving the regression problem is 0.137 s, of which 0.006 s are taken to compute the polyhedral partition through Algorithm 2.

For comparison, the same regression problem is solved through the regression algorithm of [15][3], using a linear *Support Vector Classifier* (SVC) [29] to compute the partition. The obtained BFR is 95.04 % and the CPU time needed to solve the regression problem is around 63 s (i.e. 459x slower than the method proposed in this work). To provide another element of comparison, Table 1 shows the parameters estimated through the approach of this report and of [15].

4.1.1 Convergence properties

As the accuracy of the final model estimate and the total CPU time is influenced by the number M of runs of Algorithm 1, the performance of the proposed learning approach has been tested with respect to both M and the dimension N of the considered training set. The obtained BFR as a function of iterations of Algorithm 1 is plotted in Fig. 1 for different lengths of the training dataset. Algorithm 1 converges after 2 runs.

4.1.2 Performance of multicategory discrimination algorithms

We compare the following four multicategory discrimination algorithms used to generate the partition of the regressor space against the 200-length validation dataset previously used in Section 4.2.1:

- *robust linear programming* (RLP) [10][4];

[3]The Hybrid Identification Toolbox [14] has been used.

[4]The solver *Gurobi* is used to compute the solution of the formulated linear programming

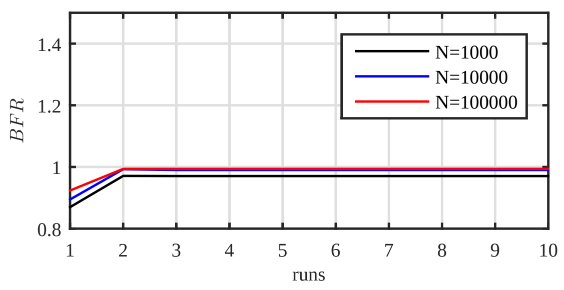


Figure 1: Example 1: BFR vs number of runs of Algorithm 1

- *regularized piecewise-smooth Newton* (RPSN) method (Algorithm 2), using the same parameters reported in Section 4.2.1;
- *averaged stochastic gradient descent* (ASGD) method (Algorithm 3), with $\lambda = 10^{-5}$ and $\nu_0 = 0.01$. The weights π_i and the initial estimate ξ_0 are computed by executing the batch Algorithm 2 on the first 50 training samples. The remaining training samples are processed recursively;
- *multicategory support vector machines* (MSVM) with linear kernels [20], implemented in the MSVMpack 1.5 toolbox [19][5].

The obtained BFRs are reported in Table 2, along with the BFR obtained when the Voronoi diagram induced by the clusters' centroids is used to partition the regressor space. The CPU time required to generate the polyhedral partition of the regressor space is given in Table 10. The performance of the MSVM approach is evaluated only in relation to the smaller training set, as larger data sets take too long to be processed. Results in Table 2 show that the first three multi-category discrimination algorithms lead to an accurate estimate of the true function in terms of output prediction, with BFRs larger then 95 % (note that a lower performance is achieved by the MSVM approach). Furthermore, the estimated models become more accurate as the number of training samples increases. This does not hold when the Voronoi diagram is used as a partition, showing that the "true" partition cannot be described by a Voronoi diagram induced by the clusters' centroids. As a matter of fact, the Voronoi diagram only depends on the clusters' centroids, and it does not take into account how the points are spread around the centroids. It is also worth noting that, for a large training set (i.e., $N = 100000$), Algorithms 2 and 3 are about 234x and 1126x faster, respectively, than the robust linear programming method of [10].

4.2 Example 2: Identification of a static three-dimensional nonlinear function

The data are generated by the following (unknown) function

$$f_o(x) = \begin{cases} h(x) & \text{if } -0.5 \leq h(x) \leq 0.5, \\ 0.5 & \text{if } h(x) \geq 0.5, \\ -0.5 & \text{if } h(x) \leq -0.5, \end{cases} \tag{22}$$

problem.

[5]The default parameters of the MSVMpack toolbox are used.

Table 2: Example 1: BFR vs length N of the training set.

	$N = 1000$	$N = 10000$	$N = 100000$
RLP [10]	97.05 %	99.59 %	99.83 %
RPSN (Algorithm 2)	97.05 %	99.01 %	99.38 %
ASGD (Algorithm 3)	94.91 %	99.07 %	99.65 %
MSVM [20]	54.46 %	–	–
Voronoi	95.74 %	86.27 %	95.62 %

Table 3: Example 1. CPU time required to partition the regressor space vs length N of the training set.

	$N = 1000$	$N = 10000$	$N = 100000$
RLP [10]	0.031 s	1.053 s	53.615 s
RPSN (Algorithm 2)	0.006 s	0.016 s	0.229 s
ASGD (Algorithm 3)	0.003 s	0.008 s	0.048 s
MSVM [20]	62.203 s	–	–

with $h : \mathbb{R}^3 \to \mathbb{R}$, $h(x) = 0.6 \sin\left(x_1 + x_2^2 - x_3\right)$. The regressor $x(k) \in \mathbb{R}^3$ is a white noise sequence with uniform distribution on the box $[-1\ 1]^3$ and length $N = 1250$. The output of the function f_o is corrupted by an additive zero-mean white noise $e_\text{o}(k) \in \mathbb{R}$, with Gaussian distribution and variance $\Lambda_e = 0.02^2$ (i.e., $y(k) = f_\text{o}(x(k)) + e_\text{o}(k)$). This corresponds to a *Signal-to-Noise Ratio* (SNR) of 25 dB.

4.2.1 Estimation results

We run Algorithm 1 with initial guess A_i as in (5) and $\Lambda_e = 1$. No forgetting factor is considered (i.e., $\kappa = 1$) and δ is set equal to 10^3. The initial guess for the cluster centroids and covariance matrices $\{R_i\}_{i=1}^s$ are computed by running an instance of Algorithm 1 without the first term in eq. (A1.1). Algorithm 1 is then run again 25 times, with the full criterion (A1.1), by initializing A_i, c_i and R_i with the output of the previous run. The clusters generated by Algorithm 1 are then separated by solving problem (11) via the Piecewise-smooth Newton method described in Algorithm 2, with parameters $K = 300$, $\sigma = 0.4$, $\lambda = 10^{-5}$, $\zeta = 10^{-5}$, $g_\text{tol} = \delta_\text{tol} = 10^{-6}$, and $\xi = 0$ as initial guess. A sensitivity analysis with respect to the tuning parameters σ and ζ is also performed. Fig. 2 shows the BFRs computed on 250 samples (not used for training) for different values of the tuning parameters. We can observe that Algorithm 2 is basically not sensitive to σ and ζ.

The number s of local affine submodels is chosen by means of cross validation. Specifically, the quality of the estimated function is assessed w.r.t. a calibration data set with 250 samples not used for training. For each value of s, the BFR is computed and, among the estimated PWA functions, we selected the one providing the largest BFR. This is achieved for $s = 12$.

The quality of the estimated PWA model is assessed w.r.t. a validation dataset of noiseless $N_\text{V} = 200$ samples. The obtained BFR is 85.19 %. The total CPU time for solving the regression problem is 3 s, of which 0.257 s are taken to compute the polyhedral partition through Algorithm 2.

For comparison, the same regression problem is solved through the regression algorithm of [15][6], using the *Proximal Support Vector Classifier* (PSVC) [16] to compute the partition. The CPU time needed to solve the regression problem is around 149 s (i.e. 49x slower than the method proposed in this paper). The obtained BFR is 53.40 %. To provide another element of comparison, Fig. 4 shows the true output y_o and the output $\hat{y}$ of the functions estimated with the method proposed in this paper and the approach in [16]. The error $y_o(k) - \hat{y}(k)$ is also plotted. For a better visualization, only the samples from time 141 to 180 are reported.

4.2.2 Convergence properties

As the accuracy of the final model estimate and the total CPU time is influenced by the number M of runs of Algorithm 1, the performance of the proposed learning approach has been tested with respect to both M and the dimension N of the considered training set. The obtained BFR as a function of iterations of Algorithm 1 is plotted in Fig. 5 for different lengths of the training dataset. Algorithm 1 converges after 15 runs.

4.2.3 Performance of multicategory discrimination algorithms

We compare the following four multicategory discrimination algorithms used to generate the partition of the regressor space against the 200-length validation dataset previously used in Section 4.2.1:

- *robust linear programming* (RLP) [10][7];
- *regularized piecewise-smooth Newton* (RPSN) method (Algorithm 2), using the same parameters reported in Section 4.2.1;

[6]The Hybrid Identification Toolbox [14] has been used.

[7]The solver *Gurobi* is used to compute the solution of the formulated linear programming problem.

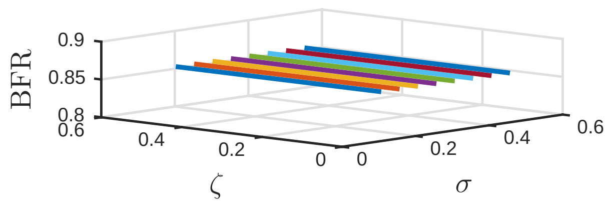


Figure 2: Example 2. BFR vs tuning parameters σ and ζ.

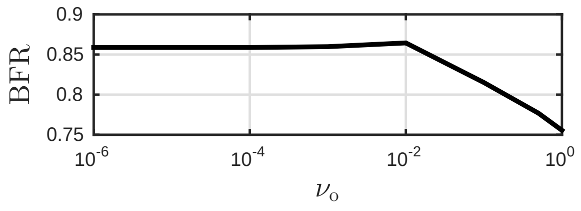


Figure 3: Example 2. BFR vs tuning parameter ν_0.

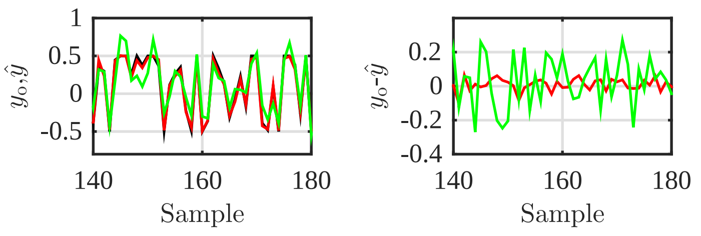


Figure 4: Example 2. Output signal (left panel): black = true, red = proposed PWA regression method, green = method in [16]. Estimation error (right panel): red = proposed PWA regression method, green = method in [16].

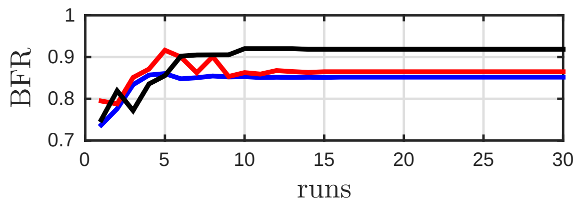


Figure 5: Example 2. BFR vs number of runs of Algorithm 1. $N = 1250$ (blue); $N = 12500$ (red); $N = 125000$ (black).

- *averaged stochastic gradient descent* (ASGD) method (Algorithm 3), with $\lambda = 10^{-5}$ and $\nu_0 = 0.01$. The weights π_i and the initial estimate ξ_0 are computed by executing the batch Algorithm 2 on the first 50 training samples. The remaining training samples are processed recursively. A sensitivity analysis w.r.t. the parameter ν_0 is also performed. Fig. 3 shows the BFRs computed for different values of ν_0. We observe that the performance of Algorithm 3 decreases as ν_o increases. This is a typical behaviour of stochastic gradient descend methods, where convergence to the global optimum improves as the parameter ν_o decreases, at the price of a lower convergence speed.

- *multicategory support vector machines* (MSVM) with linear kernels [13], implemented in the MSVMpack 1.5 toolbox [19][8].

The CPU time required to generate the polyhedral partition of the regressor space is given in Table 4. The performance of the MSVM approach is evaluated only in relation to small/medium training sets, as large data sets take too long to be processed. Notice that, for a large training set (i.e., $N = 125000$), Algorithms 2 and 3 are about 454x and 65200x faster, respectively, than the robust linear programming method of [10].

The obtained BFRs are reported in Table 5, along with the BFR obtained when the Voronoi diagram induced by the clusters' centroids is used to partition the regressor space. Results in Table 5 show that the employed multi-category discrimination algorithms lead to an accurate estimate of the true function in terms of output prediction, with BFRs larger than 80 % also in the case of small training set ($N = 1250$). This aspect indicates that the training samples are

[8]The default parameters of the MSVMpack toolbox are used.

Table 4: Example 2. CPU time required to partition the regressor space vs length N of the training set.

	$N = 1250$	$N = 12500$	$N = 125000$
RLP [10]	1.336 s	125 s	8541 s
RPSN (Algorithm 2)	0.257 s	1.762 s	18.8 s
ASGD (Algorithm 3)	0.0014 s	0.018 s	0.131 s
MSVM [20]	6.545 s	3870 s	–

Table 5: Example 2. BFR vs length N of the training set.

	$N = 1250$	$N = 12500$	$N = 125000$
RLP [10]	86.56 %	87.41 %	93.88 %
RPSN (Algorithm 2)	85.19 %	86.47 %	91.86 %
ASGD (Algorithm 3)	84.78 %	82.30 %	92.99 %
MSVM [20]	80.91 %	80.02 %	–
Voronoi	81.75 %	83.96 %	86.60 %

accurately clustered by Algorithm 1. Furthermore, the first three discrimination algorithms lead to BFRs larger than 90 % for a large training set ($N = 125000$). In the latter case, the Voronoi diagram does not achieve similar performance. This suggests that the Voronoi diagram induced by the clusters' centroids is not flexible enough in partitioning the regressor space. As a matter of fact, the Voronoi diagram only depends on the clusters' centroids, and it does not take into account how the points are spread around the centroids.

4.2.4 Monte-Carlo simulation

A Monte Carlo simulation with 100 runs, with new realizations of both the input u and the measurement noise e_o at each run, is carried out to assess the robustness of the estimation algorithm w.r.t. different realizations of the training data. The obtained results are reported in Table 6, which shows the mean and the standard deviation of the BFR over the Monte Carlo simulation for training data sets of length $N = 12500$.

4.3 Example 3: Modeling of concrete strength

The proposed PWA regression algorithm is used to estimate from experimental data a PWA function modeling the concrete compressive strength as a function of age and some of its ingredients, namely: superplasticizer content, fine aggregate content, and cement-to-water ratio. The used data set is taken from the UCI machine learning database repository [21] and it consists of 1030 records gathered from laboratory tests (see [31] for details). After shuffling the data, 730 samples are used for training and 50 samples are used to tune the number

Table 6: Example 2. Monte Carlo simulation: BFR (mean ± std).

	RPSN (Algorithm 2)	ASGD (Algorithm 3)
BFR	(87.01 ± 3.07) %	(86.88 ± 2.59) %

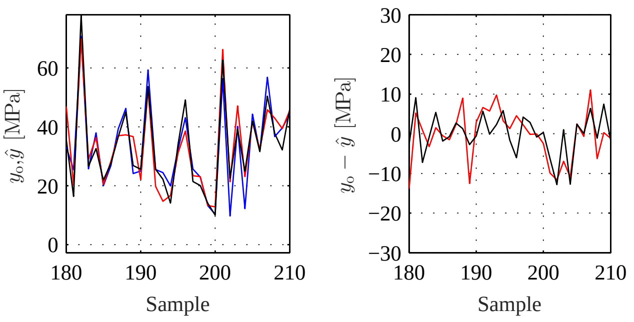


Figure 6: Example 3. Simulated output (left panel): blue = true, black = PWA regression, red = Neural Network. Simulation error (right panel): black = PWA regression, red = NN.

of affine submodels s through cross-validation. The selected value of s is 9. The performance of the estimated PWA model is tested in the remaining 250 samples.

The input/output pairs are clustered through Algorithm 1, and Algorithm 2 is used to partition the input space. The parameters and the starting conditions of the two algorithms are the same as the ones used in the previous example. For the sake of comparison, a *Neural Network* (NN) with a single hidden layer of size 15 is trained to predict the concrete compressive strength. Hyperbolic tangent sigmoid activation functions are used. The size of the hidden layer is tuned through cross-validation. The actual concrete compressive strength is plotted in Fig. 6, along with the output of the estimated PWA map and of the neural network. For the sake of visualization, only 30 out of 250 testing samples are plotted. The BFR achieved (in the testing set) by the PWA function is 52.4 %, while a BFR equal to 56.0 % is obtained with the NN. The obtained results shows that the NN achieves a slightly better accuracy than the PWA function, but this comes at the price of a more complex structure.

4.4 Example 4: Identification of a multivariable PWARX system

4.4.1 Data generating system

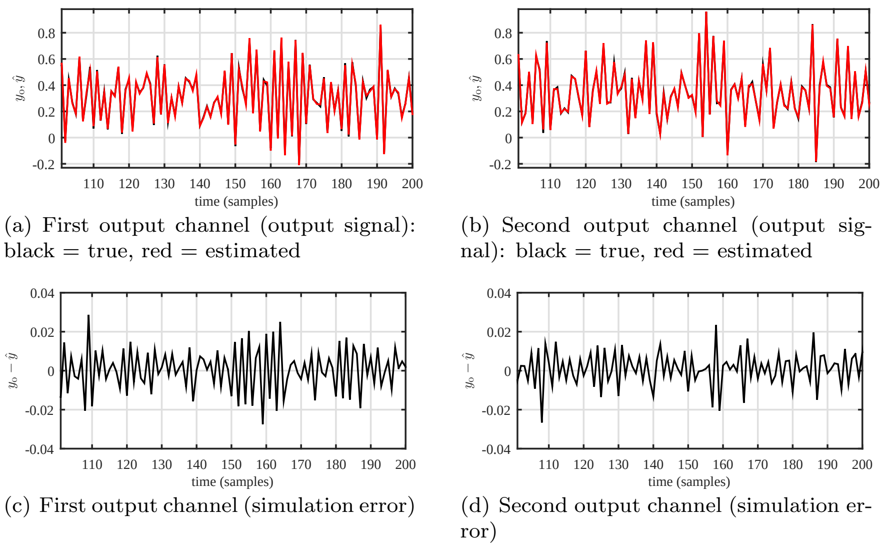
Let the system generating the data be a MIMO PWARX system described by the difference equation

$$\begin{aligned}\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} &= \begin{bmatrix} -0.83 & 0.20 \\ 0.30 & -0.52 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} + \begin{bmatrix} -0.34 & 0.45 \\ -0.30 & 0.24 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} \\ &+ \begin{bmatrix} 0.20 \\ 0.15 \end{bmatrix} + \max\left\{ \begin{bmatrix} 0.20 & -0.90 \\ 0.10 & -0.42 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} \right. \\ &\left. + \begin{bmatrix} 0.42 & 0.20 \\ 0.50 & 0.64 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + \begin{bmatrix} 0.40 \\ 0.30 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} + e_o(k). \qquad (23)\end{aligned}$$

System (23) is characterized by $\bar{s} = 4$ operating modes, given by the possible combinations of sign of the components of the first vector argument of the "max" operator. The excitation input $u(k)$ is a white noise sequence with uniform distribution in the box $[-1.0 \quad -0.4] \times [-1.0 \quad -0.4]$ and length $N = 4000$,

Table 7: PWARX identification: BFR and MSE on the two output channels

BFR_1	BFR_2	MSE_1	MSE_2
96.2%	96.3%	$0.99 \cdot 10^{-4}$	$0.70 \cdot 10^{-4}$



(a) First output channel (output signal): black = true, red = estimated

(b) Second output channel (output signal): black = true, red = estimated

(c) First output channel (simulation error)

(d) Second output channel (simulation error)

Figure 7: PWA model estimate: output signal and simulation error on the first and on the second output channel

$e_o(k) \in \mathbb{R}^2$ is a zero-mean white noise with covariance matrix $\Lambda_e = \left[\begin{smallmatrix} 0.02 & 0.02 \\ 0.02 & 0.02 \end{smallmatrix}\right]$. This corresponds to signal-to-noise ratios equal to $\text{SNR}_1 = 8.7$ dB and $\text{SNR}_2 = 6.9$ dB on the first and second output channels, respectively.

4.4.2 Identification results

We run Algorithm 1 with $s = \bar{s} = 4$ with initial guess A_i as in (5). The initial guess for the cluster centroids is computed by first running an instance of Algorithm 1 without the first term in (A1.1). Algorithm 1 is then run again for a total of 15 times, with the full criterion (A1.1), by initializing A_i, c_i with the output of the previous run. The clusters generated by Algorithm 1 are then separated via the multicategory discrimination method described in Algorithm 2. The following parameters are used in Algorithm 2: $K = 300$; $\sigma = 0.4$; $\lambda = 10^{-5}$; $\zeta = 10^{-4}$; $g_{\text{tol}} = \delta_{\text{tol}} = 10^{-6}$. As initial guess, the parameters ξ are set to zero.

The quality of the estimated PWA model is assessed w.r.t. a validation dataset of noiseless $N_V = 500$ samples. The true output y_o and the open-loop simulated output sequence $\hat{y}$ are plotted in Fig. 7, along with the simulation error $y_o(k) - \hat{y}(k)$. For the sake of visualization, only the results related to 100 samples are shown in Fig. 7. The obtained BFR and MSE are reported in Table 7. The estimated polyhedral partition of the regressor space is such that only 12 out of 500 data samples are misclassified (i.e., 2.4 % of the whole validation dataset).

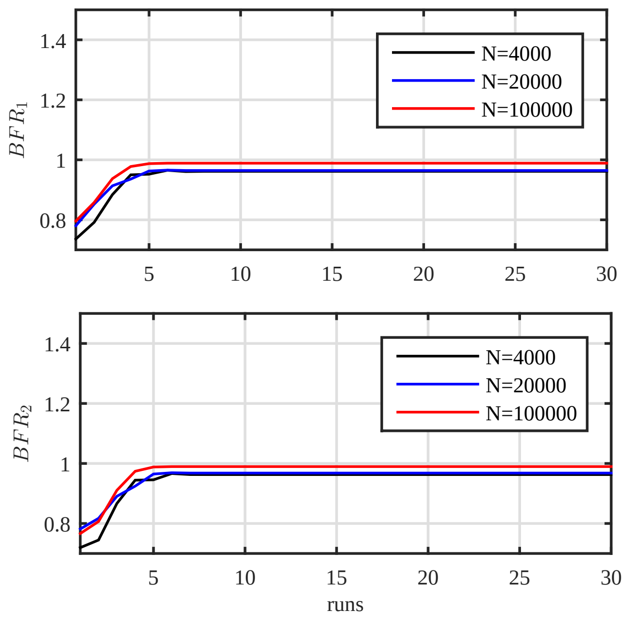


Figure 8: PWA model estimate: BFR on the first and on the output channel vs number of runs of Algorithm 1

The total CPU time for estimating the PWARX model is 0.76 s, of which 0.016 s are taken to execute Algorithm 2. For the sake of comparison, the robust linear programming approach for multicategory discrimination of [10] was also run to generate the partition, which takes 0.308 s (19x slower than Algorithm 2). An extensive comparison between the performance of Algorithm 2 and that of the classical multicategory discrimination approach of [10] is provided in Section 4.4.4.

4.4.3 Convergence properties

As the accuracy of the final model estimate and the total CPU time is influenced by the number M of runs of Algorithm 1, the performance of the proposed learning approach has been tested with respect to both M and the dimension N of the considered training set. The obtained BFR as a function of iterations of Algorithm 1 is plotted in Fig. 8 for different length of the training dataset. Clearly, there is no improvement in BFR on the two output channels after about 8 runs.

4.4.4 Performance of multicategory discrimination algorithms

This section provides a detailed comparison among the following three multicategory discrimination algorithms used to generate the partition of the regressor space:

- robust linear programming [10];
- piecewise-smooth Newton method (Algorithm 2), using the same parameters reported in Section 4.4.2;

Table 8: PWA model estimate: BFR on the two output channels vs length N of the training set.

		$N = 4000$	$N = 20000$	$N = 100000$
BFR_1	Algorithm [10]	96.0 %	96.5 %	99.0 %
	Algorithm 2	96.2 %	96.4 %	98.9 %
	Algorithm 3	86.7 %	95.0 %	96.7 %
BFR_2	Algorithm [10]	96.2 %	96.9 %	99.0 %
	Algorithm 2	96.3 %	96.8 %	99.0 %
	Algorithm 3	87.4 %	95.2 %	96.4 %

Table 9: PWA identification: percentage of misclassified validation samples.

	$N = 4000$	$N = 20000$	$N = 100000$
Algorithm [10]	1.6 %	1.0 %	0.2 %
Algorithm 2	2.4 %	1.2 %	0.4 %
Algorithm 3	5.4 %	2.0 %	1.4 %

- averaged stochastic gradient descent (Algorithm 3), with $\lambda = 10^{-5}$ and $\nu_0 = 10^{-3}$. The weights π_i and the initial estimate ξ_0 are computed by executing the batch Algorithm 2 on the first 3000 training samples. The remaining training samples are processed recursively.

The performance of the three multicategory discrimination algorithms is assessed against the 500-length validation dataset previously used in Section 4.4.2. The obtained BFR's on the two output channels are reported in Table 8, while Table 9 shows the percentage of misclassified validation data samples. The CPU time required by the three multicategory discrimination algorithms to generate the polyhedral partition of the regressor space is given in Table 10. Results in Tables 8 and 9 show that all of the three algorithms used to compute the polyhedral partition of the regressor space lead to an accurate estimate of the system, both in terms of output prediction (Table 8) and partition of the regressor space (Table 9), with BFRs larger then 95 % and a percentage of misclassified points smaller than 2.5 % (except when $N = 4000$ and Algorithm 3 is used). Furthermore, the estimated models become more accurate as the number of training samples increases. It is also worth noting that, for a large training set (i.e., $N = 100000$), Algorithms 2 and 3 are about 300x and 1600x faster, respectively, than the robust linear programming method of [10].

4.4.5 Online learning

The performance of the proposed approach is also tested in relation to an online learning problem with a dataset of $N = 100000$ samples. The generating PWARX system is supposed to change with respect to (23) after 10000 time samples. The remaining samples (i.e., 90000 data points) are assumed to be

Table 10: PWA identification: CPU time required to partition the regressor space vs length N of the training set.

	$N = 4000$	$N = 20000$	$N = 100000$
Algorithm [10]	0.308 s	3.227 s	112.435 s
Algorithm 2	0.016 s	0.086 s	0.365 s
Algorithm 3	0.013 s	0.023 s	0.067 s

generated by the following PWARX system:

$$\begin{aligned}\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} &= \begin{bmatrix} -0.8579 & 0.1672 \\ 0.2076 & -0.4962 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} \\ &+ \begin{bmatrix} -0.2958 & 0.4612 \\ -0.3085 & 0.2807 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + \begin{bmatrix} 0.2102 \\ 0.1323 \end{bmatrix} \\ &+ \max\left\{ \begin{bmatrix} 0.1884 & -0.9455 \\ 0.0808 & -0.3677 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} \right. \\ &\left. + \begin{bmatrix} 0.4486 & 0.1172 \\ 0.4093 & 0.5946 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + \begin{bmatrix} 0.4272 \\ 0.2787 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} \\ &+ e_o(k), \end{aligned} \tag{24}$$

Regarding the setup of the test, the first 3000 samples (all obtained from (23)) are preprocessed offline and used in Algorithm 1 and Algorithm 2 to determine an initial estimate of the model parameters and of the regressor space partition, respectively. The remaining 97000 data points are processed online, and the regressor space partition is recursively computed through Algorithm 3.

In order to assess the performance of the proposed approach, the difference between the true (noise-free) output $y_o(k) = y(k) - e_o(k)$ and the one-step ahead prediction of the model output $\hat{y}(k)$ is considered. In particular, the Relative Mean Square Error (RMSE) indicator

$$\text{RMSE}_i = \frac{\sum_{k=1}^{N} \left(y_{o,i}(k) - \hat{y}_i(k)\right)^2}{\sum_{k=1}^{N} y_{o,i}^2(k)}, \tag{25}$$

which provides the ratio between the mean square error and the power of the actual output signal, is computed. The obtained values of RMSE_1 and RMSE_2 are 0.8 % and 0.1 %, respectively.

As far as the computational complexity is concerned, the average CPU time required to cluster the observed regressor and update the model parameters (Algorithm 1) at each time instant is 324 μs, while Algorithm 3 requires 34 μs (in average) to update the partition of the regressor space. Based on these results, the proposed recursive estimation approach seems to be suited for adaptive control applications with sampling times down to the order of milliseconds.

4.5 Example 5: Identification of an LPV system

4.5.1 Data generating system

Let the data be collected from the MIMO LPV system

$$\begin{aligned}\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} &= \begin{bmatrix} \bar{a}_{1,1}(p(k)) & \bar{a}_{1,2}(p(k)) \\ \bar{a}_{2,1}(p(k)) & \bar{a}_{2,2}(p(k)) \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} \\ &\quad + \begin{bmatrix} \bar{b}_{1,1}(p(k)) & \bar{b}_{1,2}(p(k)) \\ \bar{b}_{2,1}(p(k)) & \bar{b}_{2,2}(p(k)) \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + e_{\mathrm{o}}(k), \end{aligned} \tag{26}$$

where

$$\begin{aligned}
\bar{a}_{1,1}(p(k)) &= \begin{cases} -0.3 & \text{if} \quad 0.4\,(p_1(k)+p_2(k)) \leq -0.3, \\ 0.3 & \text{if} \quad 0.4\,(p_1(k)+p_2(k)) \geq 0.3, \\ 0.4\,(p_1(k)+p_2(k)) & \text{otherwise}, \end{cases} \\
\bar{a}_{1,2}(p(k)) &= 0.5\,(|p_1(k)|+|p_2(k)|)\,, \\
\bar{a}_{2,1}(p(k)) &= p_1(k)-p_2(k), \\
\bar{a}_{2,2}(p(k)) &= \begin{cases} 0.5 & \text{if} \quad p_1(k) < 0, \\ 0 & \text{if} \quad p_1(k) = 0, \\ -0.5 & \text{if} \quad p_1(k) > 0, \end{cases} \\
\bar{b}_{1,1}(p(k)) &= 3p_1(k)+p_2(k), \\
\bar{b}_{1,2}(p(k)) &= \begin{cases} 0.5 & \text{if} \quad 2\left(p_1^2(k)+p_2^2(k)\right) \geq 0.5, \\ 2\left(p_1^2(k)+p_2^2(k)\right) & \text{otherwise}, \end{cases} \\
\bar{b}_{2,1}(p(k)) &= 2\sin\{p_1(k)-p_2(k)\}\,, \\
\bar{b}_{2,2}(p(k)) &= 0.
\end{aligned}$$

Both the input $u(k)$ and the scheduling variable $p(k)$ are white noise sequences (independent of each other) of length $N = 11000$ with uniform distribution in the boxes $[-0.5 \;\; 0.5] \times [-0.5 \;\; 0.5]$ and $[-1 \;\; 1] \times [-1 \;\; 1]$, respectively. The noise covariance matrix of $e_{\mathrm{o}}(k) \in \mathbb{R}^2$ is $\Lambda_e = \left[\begin{smallmatrix} 0.25 & 0 \\ 0 & 0.25 \end{smallmatrix}\right]$. This corresponds to signal-to-noise ratios on the first and on the second output channel equal to $\mathrm{SNR}_1 = 4$ dB and $\mathrm{SNR}_2 = 7$ dB, respectively.

4.5.2 Choice of the number of modes

The number s of polyhedral regions defining the partition of the scheduling variable space $\mathcal{P} = [-1 \;\; 1] \times [-1 \;\; 1]$ is chosen by means of cross validation. We split the 11000-length training data set into two disjoint sets. The first 10000 samples are used to estimate a PWA approximation of the true nonlinear functions $\bar{a}_{i,j}$ and $\bar{b}_{i,j}$, along with the polyhedral partition of the scheduling variable space $\mathcal{P}$, for different values of s in the range 5–30. For each value of s, the identification Algorithm 1 is run 10 times. The second part of training data (the remaining 1000 samples) is used to assess the quality of the identified LPV models. Specifically, for each value of s, the BFR on the two output channels is computed. The obtained BFR_i (with $i = 1, 2$), as a function of s, are plotted in Fig. 9. Among the identified LPV models, the one providing the largest aggregated $\mathrm{BFR}_{\mathrm{T}} = \mathrm{BFR}_1 + \mathrm{BFR}_2$ is selected, which corresponds to $s = 10$ polyhedral regions. The computed polyhedral partition is plotted in Fig. 10 (the Hybrid Toolbox for MATLAB [5] has been used to plot the polytopes in Fig. 10).

Table 11: LPV identification: validation results in terms of BFR and MSE of the LPV models estimated by using the PWA regression approach of Algorithms 1, 2 and the parametric LPV identification approach of [4]

	BFR_1	BFR_2	MSE_1	MSE_2
PWA regression	87 %	84 %	$6.9{\cdot}10^{-3}$	$13.9{\cdot}10^{-3}$
parametric LPV	80 %	70 %	$17.5{\cdot}10^{-3}$	$46.3{\cdot}10^{-3}$

4.5.3 Model quality assessment

The quality of the estimated LPV model is assessed w.r.t. a validation dataset, consisting of a new sequence of $N_{\text{V}} = 2000$ noiseless samples used neither to estimate the LPV model nor to select the number of modes s. For the sake of comparison, the nonlinear coefficient functions $\bar{a}_{i,j}(p(k))$ and $\bar{b}_{i,j}(p(k))$ are also estimated through the parametric LPV identification approach proposed in [4], by parameterizing the nonlinear functions $\bar{a}_{i,j}(p(k))$ and $\bar{b}_{i,j}(p(k))$ as fourth-order polynomials in the two-dimensional scheduling variable $p(k)$.

The true outputs y_{o} and the simulated output sequences $\hat{y}$ of the estimated LPV models are plotted in Fig. 11, along with the simulation error $y_{\text{o}}(k) - \hat{y}(k)$. For the sake of visualization, only the samples from time 101 to 200 are reported. The BFR and MSE on the two output channels are reported in Table 11. The obtained results show that the proposed LPV identification approach based on the PWA approximation of the coefficient functions $\bar{a}_{i,j}(p(k))$ and $\bar{b}_{i,j}(p(k))$) outperforms the parametric LPV identification approach in [4], which in turn is based on a priori specified polynomial parametrization of the nonlinear functions $\bar{a}_{i,j}(p(k))$ and $\bar{b}_{i,j}(p(k))$.

We remark that the "online" computational time required to evaluate the output of the LPV model, given the current value of the scheduling variable $\bar{p}$ and the past input/output observations is about 120 μs, 40 μs of which are required to evaluate which region the current scheduling variable belongs to. Note that the latter step requires to compute the maximum of $s = 10$ affine functions $\{\phi_i(\bar{p})\}_{i=1}^{s}$ defining the piecewise affine separator $\phi(\bar{p})$ in (6). This relatively "low" online computational time is mainly due to the PWA structure

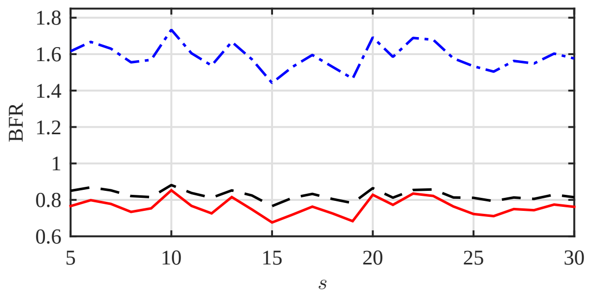


Figure 9: LPV identification: BFR_1 (black dashed line), BFR_2 (red solid line) and aggregate $\text{BFR}_{\text{T}} = \text{BFR}_1 + \text{BFR}_2$ (blue dashed-dot line) vs number of modes (s).

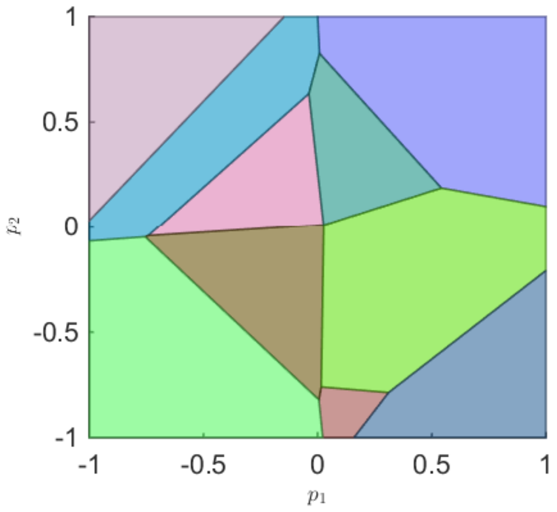


Figure 10: LPV identification: polyhedral partition of the scheduling variable space $\mathcal{P}$

of the coefficient functions describing the LPV model and it allows to use the estimated LPV model in applications requiring a "fast" online determination of the operating mode, such as in gain scheduling or in LPV model predictive control.

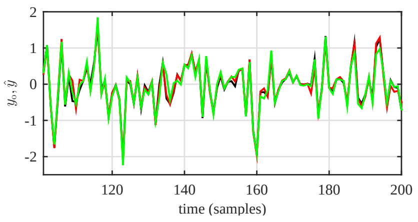
4.5.4 Performance of multicategory discrimination algorithms

The CPU time required to estimate the LPV model through the proposed PWA regression approach is 759 s. This includes the cross-validation phase to compute the number of modes s. For $s = 10$, the CPU time required to compute the LPV model is 14 s, 0.4 s of which are spent to compute the polyhedral partition via Algorithm 2 (the robust linear programming multicategorical discrimination algorithm of [10] takes 4.2 seconds, that is almost 10x slower).

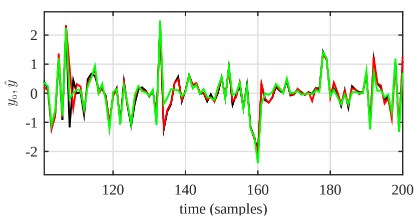
For a more exhaustive comparison between the Newton-like approach (Algorithm 2) and the robust linear programming algorithm of [10], the CPU time required by the two algorithms to partition the scheduling parameter space is plotted, as a function of s, in Fig. 12. Fig. 12 also shows the CPU time required by the averaged stochastic gradient descent algorithm (Algorithm 3) to solve the same problem. The weights π_i and the initial estimate ξ_0 for Algorithm 3 are computed by executing the batch Algorithm 2 on the first 1000 training samples. The remaining 9000 training samples are processed recursively. The CPU time reported in Fig. 12 also includes the time required to initialize Algorithm 3. The chosen parameters for Algorithm 2 and 3 are:

- Algorithm 2: $K = 300$; $\sigma = 0.4$; $\lambda = 10^{-5}$; $\zeta = 10^{-4}$; $g_{\mathrm{tol}} = \delta_{\mathrm{tol}} = 10^{-6}$; initial guess for $\xi : \mathbf{0}_n$;
- Algorithm 3: $\lambda = 10^{-5}$ and $\nu_0 = 10^{-3}$.

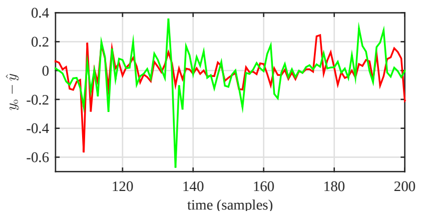
In order to test the performance of the three multicategory discrimination



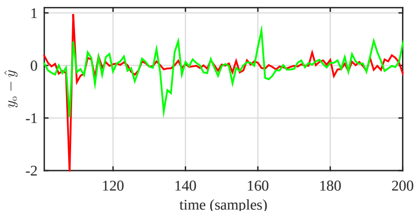
(a) First output channel (output signal): black = true, red = PWA regression, green = polynomial parameterization



(b) Second output channel (output signal): black = true, red = PWA regression, green = polynomial parameterization



(c) First output channel (simulation error): red = PWA regression, green = polynomial parameterization



(d) Second output channel (simulation error): red = PWA regression, green = polynomial parameterization

Figure 11: LPV model estimate: output signal and simulation error on the first and on the second output channel. Estimate obtained through the PWA regression approach (red lines); estimate obtained through the parametric approach [4] (green lines)

algorithms in terms of model accuracy, the aggregate best fit rate BFR_T obtained by using the three algorithms is plotted, as a function of s, in Fig. 13. Results in Figs. 12 and 13 show that:

- the CPU time required by all of the three discrimination algorithms to partition the scheduling variable space increases with the number of modes s (Fig. 12). This is due to the fact that the number of parameters ξ defining the piecewise affine separator $\phi(x)$ in (6) increases linearly with s;
- the two multicategory discrimination algorithms presented in this work (i.e., Algorithm 2 and Algorithm 3) are faster (from 6x to 20x) than the robust linear programming based approach of [10].
- in terms of model accuracy, the robust linear programming approach of [10] and the Newton-like method of Algorithm 2 achieve similar performance (Fig. 13), while, for $s = 11$, $s = 14$ and $s = 20$, the averaged stochastic gradient descent algorithm (Algorithm 3) does not provide an accurate partition of the scheduling variable space, leading to LPV models with an aggregate best fit rates smaller than 1.1. This means that, for $s = 11, 14, 20$ the solution of the averaged stochastic gradient descent algorithm fails to converge to the batch solution of Problem (11) when only $N = 10000$ training samples are used.

5 Conclusions

The strengths of the PWA regression approach of this work are (*i*) its computational efficiency, (*ii*) the ability to be run both in a batch and in a recursive way, and (*iii*) the quality of fit that can be achieved. Future research will be devoted to generalize the approach to piecewise-nonlinear models (such as piecewise polynomial) by feeding regression data manipulated through nonlinear basis functions to Algorithms 1, 2 and 3.

References

[1] S.T. Alexander and A.L. Ghirnikar. A method for recursive least squares filtering based upon an inverse QR decomposition. *IEEE Trans. Signal Processing*, 41(1):20–30, 1993.

[2] P. Apkarian and P. Gahinet. A convex characterization of gain-scheduled $\mathcal{H}_\infty$ controllers. *IEEE Transactions on Automatic Control*, 40(5):853–864, 1995.

[3] L. Bako, K. Boukharouba, E. Duviella, and S. Lecoeuche. A recursive identification algorithm for switched linear/affine models. *Nonlinear Analysis: Hybrid Systems*, 5(2):242–253, 2011.

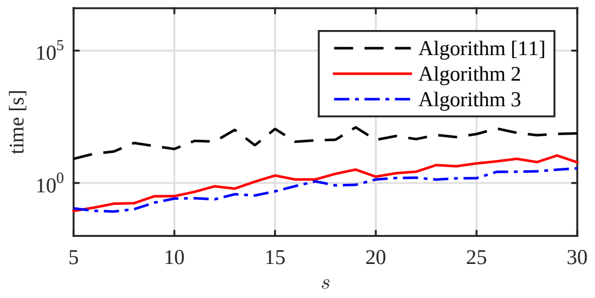


Figure 12: LPV identification: CPU time required to partition the scheduling variable space vs number of modes (s).

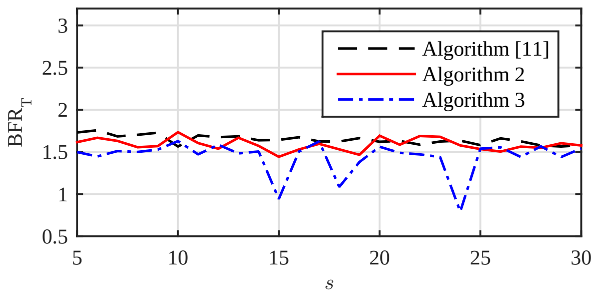


Figure 13: LPV identification: aggregate best fit rate $\mathrm{BFR_T}$ vs number of modes (s)

[4] B. A. Bamieh and L. Giarré. Identification of linear parameter-varying models. *International Journal of Robust Nonlinear Control*, 12(9):841–853, 2002.

[5] A. Bemporad. Hybrid Toolbox - User's Guide, 2004. `http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox`.

[6] A. Bemporad, D. Bernardini, and P. Patrinos. A convex feasibility approach to anytime model predictive control. Technical report, IMT Lucca, February 2015. `http://arxiv.org/abs/1502.07974`.

[7] A. Bemporad, V. Breschi, and D. Piga. Piecewise affine regression via recursive multiple least squares and multicategory discrimination. Technical report, IMT Institute for Advanced Studies, Lucca, October 2015. Submitted to Automatica. Available at: `http://www.dariopiga.com/TR/TR_PWAReg_BBP_2015.pdf`.

[8] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Transactions on Automatic Control*, 50(10):1567–1580, 2005.

[9] A. Bemporad, M. Morari, and N.L. Ricker. *Model Predictive Control Toolbox for MATLAB 5.0*. The Mathworks, Inc., 2015. `http://www.mathworks.com/help/mpc/ug/adaptive-mpc.html`.

[10] K.P. Bennett and O.L. Mangasarian. Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3:27–39, 1994.

[11] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[12] V. Breschi, A. Bemporad, and D. Piga. Identification of hybrid and linear parameter varying models via recursive piecewise affine regression and discrimination. Technical report, IMT Lucca, October 2015. Submitted to the European Control Conference 2016. `http://www.dariopiga.com/TR/TR_PWAReg_BBP_ECC_2016.pdf`.

[13] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

[14] G. Ferrari-Trecate. Hybrid Identification Toolbox (HIT), 2005.

[15] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

[16] G.M. Fung and O.L. Mangasarian. Multicategory proximal support vector machine classifiers. *Machine Learning*, 59:77–97, 2005.

[17] A. Garulli, S. Paoletti, and A. Vicino. A survey on switched and piecewise affine system identification. In *16th IFAC Symposium on System Identification*, pages 344–355, Brussels, Belgium, 2012.

[18] A. L. Juloski, S. Weiland, and W. P. M. H. Heemels. A bayesian approach to identification of hybrid systems. *IEEE Transactions on Automatic Control*, 50(10):1520–1533, 2005.

[19] F. Lauer and Y. Guermeur. MSVMpack: a multi-class support vector machine package. *Journal of Machine Learning Research*, 12:2269–2272, 2011. `http://www.loria.fr/~lauer/MSVMpack`.

[20] Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *J. American Statistical Association*, 99:67–81, 2004.

[21] M. Lichman. UCI machine learning repository, 2013.

[22] H. Nakada, K. Takaba, and T. Katayama. Identification of piecewise affine systems based on statistical clustering technique. *Automatica*, 41(5):905–913, 2005.

[23] A. Packard. Gain scheduling via linear fractional transformations. *Systems & Control Letters*, 22(2):79–92, 1994.

[24] S. Paoletti, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal. Identification of hybrid systems a tutorial. *European journal of control*, 13(2):242–260, 2007.

[25] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

[26] D. Ruppert. Efficient estimations from a slowly convergent Robbins-Monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

[27] C. W. Scherer. Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control for time-varying and linear parametrically-varying systems. *Int. Journal of Robust and Nonlinear Control*, 6(9-10):929–952, 1996.

[28] R. Tóth. *Modeling and identification of linear parameter-varying systems.* Lecture Notes in Control and Information Sciences, Vol. 403, Springer, Heidelberg, 2010.

[29] V. Vapnik. *Statistical learning theory.* Wiley New York, 1998.

[30] W. Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*, 2011.

[31] I. C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.