Piecewise Affine Regression via Recursive Multiple Least Squares and Multicategory Discrimination ⋆

Valentina Breschi [a], Dario Piga [a], Alberto Bemporad [a].

[a] *IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy.*

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 paper, we propose a novel and 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*) 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.

Key words: PWA regression; System identification; Clustering; Recursive multiple least squares; Multicategory discrimination.

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. Piece-Wise Affine (PWA) 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 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 is an NP-hard problem in general (see [15] for a detailed analysis on the complexity of PWA regression), and several algorithms to estimate PWA maps from data are available in the literature (see [13,20] for an overview). A convex relaxation, based on ℓ_1 regularization, is proposed in [19] to approximate the underlying combinatorial problem arising from PWA regression. In [21] 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 [5, 11, 14, 18] 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 [5] 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 [18]

⋆ Corresponding author Dario Piga.

⋆⋆This work was partially supported by the European Commission under project H2020-SPIRE-636834 "DISIRE - Distributed In-Situ Sensors Integrated into Raw Material and Energy Feedstock" (`http://spire2030.eu/disire/`).

Email addresses: `valentina.breschi@imtlucca.it` (Valentina Breschi), `dario.piga@imtlucca.it` (Dario Piga), `alberto.bemporad@imtlucca.it` (Alberto Bemporad).

can become inefficient in case of PWA maps with many parameters. In [14], 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 [11], the regressor vectors are clustered through a K-means-like algorithm and the submodel parameters are obtained via weighted least-squares. Although [11] 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 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 [2], where training samples are clustered iteratively and model parameters are updated through recursive least-squares. 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 linear separation maps that may limit regression capabilities.

This paper describes a novel approach for approximating vector-valued, and possibly discontinuous, functions in PWA form, trying to overcome the aforementioned limitations of existing methods. 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 suitable for online learning, as shown in the example. The application of the proposed PWA regression algorithm to the identification of linear parameter-varying and hybrid dynamical models is discussed in [8].

The paper is organized as follows. The PWA regression problem is described in Section 2. 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. A simulation example is reported in Section 4 to show the effectiveness of the proposed approach.

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. 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

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

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 $\mathcal{X}_i$, $i = 1, \dots, s$ 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 trade-off between fitting the data and avoiding model complexity and overfit, with consequent poor generalization

[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 $\mathcal{X}_i$.

on unseen data. This is related to one of the most crucial aspects in function learning, known as *bias-variance tradeoff* [23]. In this work, we assume that s is fixed by the user. The value of s can be chosen through cross-validation, with a possible upper-bound dictated by the maximum tolerable complexity of the estimated model.

3 PWA regression algorithm

As mentioned in Section 1, we tackle the PWA regression problem 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

$$A_i \equiv \arg\min_{A} \sum_{k=1}^{N} \|y(k) - A \begin{bmatrix} 1 \\ x(k) \end{bmatrix}\|_2^2, \ \forall i = 1, \ldots, s \quad (2)$$

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 inverse of the cluster covariance matrices $R_i = \dfrac{1}{|\mathcal{C}_i| - 1} \displaystyle\sum_{x(k) \in \mathcal{C}_i} [x(k) - c_i]\,[x(k) - c_i]'$.

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

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** $e_i(k) \leftarrow y(k) - A_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix}$, $i = 1, \ldots, s$;
   - 3.2. **let**

$$i(k) \leftarrow \arg\min_{i=1,\ldots,s} (x(k) - c_i)' R_i^{-1} (x(k) - c_i) + e_i(k)' \Lambda_e^{-1} e_i(k); \quad \text{(A1.1)}$$

   - 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 inverse of the cluster covariance matrix $R_{i(k)}$ for cluster $\mathcal{C}_{i(k)}$ through the Matrix Inversion Lemma:

$$Q \leftarrow R_{i(k)}^{-1} - \frac{R_{i(k)}^{-1} \left[x(k) - c_{i(k)}\right] \left[x(k) - c_{i(k)}\right]' R_{i(k)}^{-1}}{|\mathcal{C}_{i(k)}| - 2 + \left[x(k) - c_{i(k)}\right]' R_{i(k)}^{-1} \left[x(k) - c_{i(k)}\right]};$$

$$R_{i(k)}^{-1} \leftarrow \frac{|\mathcal{C}_{i(k)}| - 1}{|\mathcal{C}_{i(k)}| - 2} \left(Q - \frac{Q \delta c_{i(k)} \delta c_{i(k)}' Q}{\frac{|\mathcal{C}_{i(k)}| - 2}{|\mathcal{C}_{i(k)}| - 1} + \delta c_{i(k)}' Q \delta c_{i(k)}} \right);$$

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 $\{R_i\}_{i=1}^{s}$.

execution as the sample covariance matrix

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

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

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 [2] for online PWA regression. However, [2] 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 $\ell = 1, \ldots, n_y$. 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.,

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

Further suppose that the residual $e(k) = y(k) - A_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix}$ 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:

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

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

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

3.2 Partitioning the regressor space

We propose here a variation of the multicategory discrimination technique of [6] to separate the clusters $\{\mathcal{C}_i\}_{i=1}^s$ that partition the regressor space in a much more computationally efficient way, especially when dealing with a large number N of data points.

For $i = 1, \ldots, s$, 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 $\mathcal{C}_1, \ldots, \mathcal{C}_s$. The piecewise affine separator ϕ is defined as the maximum of s affine functions $\{\phi_i(x)\}_{i=1}^s$, i.e.,

$$\phi(x) = \max_{i=1,\ldots,s} \phi_i(x), \tag{3}$$

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

$$\mathcal{X}_i = \{x \in \mathbb{R}^{n_x} : \phi_i(x) = \phi(x)\}. \tag{4}$$

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:

$$\phi_i(x) = \begin{bmatrix} x' & -1 \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix}. \tag{5}$$

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

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

or, equivalently,

$$\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} \geq \begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + \mathbf{1}_{m_i}, \quad i,j = 1,\dots,s,\ i \neq j, \tag{6}$$

where the constant vector $\mathbf{1}_{m_i}$ on the right side of eq. (6) is used only for normalization purposes.

A piecewise-affine separator ϕ thus satisfies the conditions:

$$\begin{cases} \phi(x) = \begin{bmatrix} x' & -1 \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix},\ \forall x \in \mathcal{C}_i,\ \ i = 1,\dots,s \\ \phi(x) \geq \begin{bmatrix} x' & -1 \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + 1,\ \forall x \in \mathcal{C}_i,\ i \neq j \end{cases} \tag{7}$$

Remark 1 According to the definition of ϕ (eq. (3)) and ϕ_i (eq. (5)), and the conditions in (7), the polyhedra $\{\mathcal{X}_i\}_{i=1}^s$ are defined as

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

■

Rather than solving a linear program as in [6], we determine $\{\omega^i, \gamma^i\}_{i=1}^s$ by solving the convex unconstrained optimization problem

$$\min_{\{\omega^i,\gamma^i\}_{i=1}^s} \frac{\lambda}{2} \sum_{i=1}^{s} \left(\|\omega^i\|_2^2 + (\gamma^i)^2 \right) + \sum_{i=1}^{s} \sum_{\substack{j=1 \\ j \neq i}}^{s} \frac{1}{m_i} \left\| \left(\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j - \omega^i \\ \gamma^j - \gamma^i \end{bmatrix} + \mathbf{1}_{m_i} \right)_+ \right\|_2^2, \tag{8}$$

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 (8) and to guarantee that (8) has a unique solution. Furthermore, by tuning the hyper-parameter λ through cross-validation, the ℓ_2-regularization term may lead to an improvement of the generalization performance of the final separator ϕ.

Problem (8) generates a piecewise-affine function that minimizes the (averaged) squared 2-norm of the violation of the inequalities (6). The problem is solved by using a regularized piecewise-smooth Newton method with Armijo's line search similar to the one proposed in [3] for functions $g : \mathbb{R}^{n_\xi} \to \mathbb{R}$ of the form

$$g(\xi) = \frac{\lambda}{2} \|\xi\|_2^2 + \sum_{j=1}^{n_g} \|g_j(\xi)_+\|_2^2, \tag{9}$$

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 (8), 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:

$$g_j(\xi) = a_j' \xi - b_j,\ j = 1,\dots,n_g, \tag{10}$$

where $n_g = N(s-1)$ and $a_j \in \mathbb{R}^{n_\xi}$, $b_j \in \mathbb{R}$ are easily obtained from (8) as a function of matrices $\{M_i\}_{i=1}^s$ and coefficients $\{m_i\}_{i=1}^s$. By letting

$$\mathcal{A} = [a_1 \ \dots \ a_{ng}]',\ \mathcal{B} = [b_1 \ \dots b_{n_g}]', \tag{11}$$

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,

$$g(\xi) = \frac{\lambda}{2} \xi' \xi + \sum_{i \in I(\xi)} (\mathcal{A}_i \xi - \mathcal{B}_i)^2 \tag{12a}$$

$$\nabla g(\xi) = \lambda \xi + \mathcal{A}_{I(\xi)}' (\mathcal{A}_{I(\xi)} \xi - \mathcal{B}_{I(\xi)}) \tag{12b}$$

$$\nabla^2 g(\xi) = \lambda I + \mathcal{A}_{I(\xi)}' \mathcal{A}_{I(\xi)} = \lambda I + \sum_{i \in I(\xi)} \mathcal{A}_i' \mathcal{A}_i \tag{12c}$$

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

The proposed approach to solve (8) is summarized in Algorithm 2. The algorithm uses the solution d of the linear system

$$(\nabla^2 g(\xi) + \delta(\xi) I) d = -\nabla g(\xi) \tag{13}$$

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 (12c), the linear system (13) is solved at Steps 5.1–5.2 as the least squares problem

$$\min_d \frac{1}{2} \left\| \begin{bmatrix} \mathcal{A}_{I(\xi)} \\ \sqrt{\lambda + \delta(\xi)} I_{n_\xi} \end{bmatrix} d + \begin{bmatrix} \mathcal{A}_{I(\xi)} \xi - \mathcal{B}_{I(\xi)} \\ \frac{\lambda}{\sqrt{\lambda + \delta(\xi)}} \xi \end{bmatrix} \right\|_2^2 \tag{14}$$

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

Algorithm 2 Piecewise-smooth Newton method for solving the multicategory discrimination problem (8)

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 (10)–(11), $j = 1, \ldots, n_g$;
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 $\left[\begin{array}{c} \mathcal{A}_I \\ \sqrt{\lambda+\delta} I_{n_\xi} \end{array}\right]$;
   5.2. **solve** the upper-triangular linear system
   $$R_{\{1,\ldots,n_\xi\}} d = -\,(Q_{\{1,\ldots,|I|\},\{1,\ldots,n_\xi\}})' c_I - \frac{\lambda}{\lambda+\delta}(Q_{\{|I|+1,\ldots,|I|+n_\xi\},\{1,\ldots,n_\xi\}})'\xi; \quad \text{(A2.1)}$$
   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** ω^i, γ^i, $i = 1, \ldots, s$, from the solution ξ;
7. **end**.

Output: Coefficients ω^i, γ^i, $i = 1, \ldots, s$ defining the piecewise affine separator ϕ in (3)–(5).

number of iterations and prevents excessive regularization in (13) 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 with respect to the piecewise affine multicategory discrimination function minimizing only the squared 2-norm of the violation of the inequalities (6), 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 online based on streaming data, we introduce a recursive approach to solve problem (8) 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 (8)–(9) can be generalized to the following unconstrained convex stochastic optimization problem

$$\xi^* = \arg\min_\xi E_{x \in \mathbb{R}^{n_x}}\left[\ell(x, \xi)\right] + \frac{\lambda}{2}\|\xi\|_2^2 \qquad (15)$$

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

with $E_x[\cdot]$ denoting the expected value w.r.t. x. The solution of problem (15) provides a piecewise affine multicategory discrimination function satisfying (3)–(5). This aims at violating the least, on average over x, the condition in (6) 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 (15) 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 (15) by online convex optimization and, in particular, by the averaged stochastic gradient descent method of [22] as proposed in [7] (cf. also [24]), whose application to the linear multicategory discrimination problem (15) is described in Algorithm 3.

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 possibly updated while Algorithm 3 is running. Numerical experiments have shown that constant and uniform coefficients $\pi = \frac{1}{s}$ work equally well.

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

Algorithm 3 Averaged stochastic gradient descent algorithm for solving the linear multicategory discrimination problem (15)

Input: Regressor flow $x(0), x(1), \ldots$; cluster assignment function $i : \mathbb{R}^{n_x} \to \{1, \ldots, s\}$; ℓ_2-regularization parameter $\lambda \geq 0$; 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**

$$\frac{\partial \ell(\xi_k, x_k)}{\partial \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix}} \leftarrow \lambda \begin{bmatrix} \omega_k^j \\ \gamma_k^j \end{bmatrix} + \frac{1}{\pi_{i(x_k)}} \times$$

$$\begin{cases} \sum\limits_{j \in I_k} \left(x_k'(\omega_k^j - \omega_k^{i(x_k)}) - \gamma_k^j + \gamma_k^{i(x_k)} + 1 \right) \begin{bmatrix} -x_k \\ 1 \end{bmatrix} \\ \qquad \text{if } j = i(x_k) \\ \left(x_k'(\omega_k^j - \omega_k^{i(x_k)}) - \gamma_k^j + \gamma_k^{i(x_k)} + 1 \right) \begin{bmatrix} x_k \\ -1 \end{bmatrix} \\ \qquad \text{if } j \neq i(x_k), \; j \in I_k \\ 0 \text{ otherwise.} \end{cases}$$

   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** ω_k^i, γ_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 (3)–(5) at each step $k = 0, 1, \ldots$.

4 Simulation example

In this section, we show the effectiveness of the proposed PWA regression algorithm in a simulation example. Further applications of the proposed method, including the identification of linear parameter-varying and hybrid dynamical models, can be found in [4] and [8].

All computations are carried out on an i7 2.40-GHz Intel core processor with 4 GB of RAM running MATLAB R2014b. In validating the obtained models on a data sequence not used for training, we will denote by y_o and $\hat{y}$ the vector stacking the measured and estimated outputs, respectively, and by $\bar{y}_o$ the vector staking the sample mean of the measured output. The *Best Fit Rate* (BFR) indicator $\text{BFR} = \max\left\{1 - \frac{\|y_o - \hat{y}\|_2}{\|y_o - \bar{y}_o\|_2}, 0\right\} \cdot 100\%$ is used to assess model quality.

The data are generated by the (unknown) function

$$f_o(x) = \begin{cases} h(x) & \text{if } -0.5 \leq h(x) \leq 0.5, \\ 0.5 & \text{if } h(x) \geq 0.5, \\ -0.5 & \text{if } h(x) \leq -0.5, \end{cases} \tag{16}$$

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 $e_o(k) \in \mathbb{R}$, with Gaussian distribution and variance $\Lambda_e = 0.02^2$ (i.e., $y(k) = f_o(x(k)) + e_o(k)$). This corresponds to a *Signal-to-Noise Ratio* (SNR) of 25 dB, where $\text{SNR} = 10 \log \frac{\sum_{k=1}^N \left(y(k) - e_o(k)\right)^2}{\sum_{k=1}^N e_o^2(k)}$.

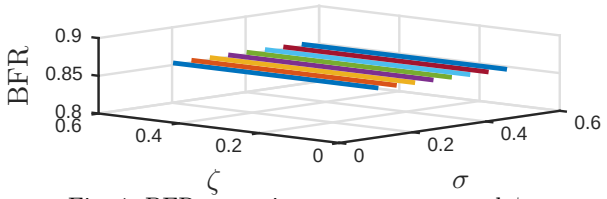

Fig. 1. BFR vs tuning parameters σ and ζ.

4.1 Estimation results

We run Algorithm 1 not considering the forgetting factor (i.e., $\kappa = 1$) and with δ set equal to 10^3. The initial guess for the parameters A_i, 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), with $\Lambda_e = 1$ and A_i initialized as in (2). 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 (8) 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. 1 shows the BFRs computed on 250 samples (not used for training) for different values of the tuning parameters. We 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 on 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 on a validation dataset with $N_V = 200$ 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 [11] [3], using the *Proximal Support Vector Classifier* (PSVC) [12]

[3] The Hybrid Identification Toolbox [10] has been used.

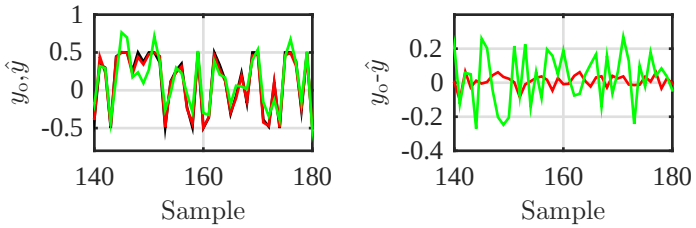

Fig. 2. Output signal (left panel): black = true, red = proposed PWA regression method, green = method in [12]. Estimation error (right panel): red = proposed PWA regression method, green = method in [12].

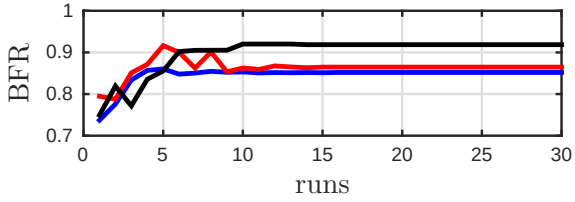

Fig. 3. BFR vs number of runs of Algorithm 1. $N = 1250$ (blue); $N = 12500$ (red); $N = 125000$ (black).

to compute the partition [4]. 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. 2 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 [12]. 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 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. 3 for different lengths of the training dataset. Algorithm 1 converges after 15 runs.

4.3 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 4.1:

- *robust linear programming* (RLP) [6] [5];

[4] Among all the classifiers available in HIT, the PSVC is the one which provided the best results.

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

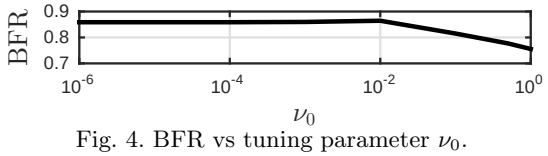

Fig. 4. BFR vs tuning parameter ν_0.

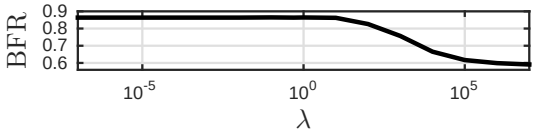

Fig. 5. Algorithm 2. BFR vs regularization parameter λ.

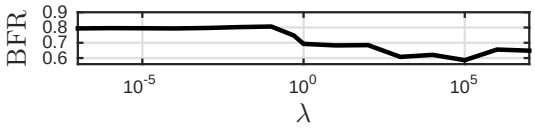

Fig. 6. Algorithm 3. BFR vs regularization parameter λ.

- *regularized piecewise-smooth Newton* (RPSN) method (Algorithm 2), using the same parameters reported in Section 4.1;
- *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. A sensitivity analysis w.r.t. the parameter ν_0 is also performed. Fig. 4 shows the BFRs computed for different values of ν_0, pointing out that the performance of Algorithm 3 decreases as ν_0 increases. This is a typical behaviour of stochastic gradient descend methods, where convergence to the global optimum improves as the parameter ν_0 decreases, at the price of a lower convergence speed.
- *multicategory support vector machines* (MSVM) with linear kernels [9], implemented in the MSVMpack 1.5 toolbox [16] [6].

A sensitivity analysis w.r.t. the regularization parameter λ is also performed. Fig. 5 and Fig. 6 show the BFRs obtained for different values of λ by Algorithms 2 and 3, respectively. Note that, for $\lambda \leq 0.1$, the final estimate is fairly insensitive to the regularization parameter λ.

The CPU time required to generate the polyhedral partition of the regressor space is given in Table 1. 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 [6].

The obtained BFRs are reported in Table 2, along with the BFR obtained when the Voronoi diagram induced

[6] The default parameters are used.

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

	$N = 1250$	$N = 12500$	$N = 125000$
RLP [6]	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 [17]	6.545 s	3870 s	–

Table 2
BFR vs length N of the training set.

	$N = 1250$	$N = 12500$	$N = 125000$
RLP [6]	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 [17]	80.91 %	80.02 %	–
Voronoi	81.75 %	83.96 %	86.60 %

Table 3
Monte Carlo simulation: BFR (mean $\pm$ std).

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

by the clusters' centroids (given as an output by Algorithm 1) is used to partition the regressor space. Results in Table 2 show that the employed 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 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.

4.4 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 3, which shows the mean and the standard deviation of the BFR over the Monte Carlo simulation for training data sets of length $N = 12500$.

5 Conclusions

The strengths of the PWA regression approach of this paper 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.

Acknowledgments

The authors would like to thank Dr. Dimitar Filev for his valuable comments to improve the contribution of the paper.

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] 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.

[3] 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`.

[4] A. Bemporad, V. Breschi, and D. Piga. Piecewise Affine Regression via Recursive Multiple Least Squares and Multicategory Discrimination. Technical report, TR-IMT-DYSCO-2016-01, 2016. Available online at: `http://www.dariopiga.com/TR/TR-IMT-DYSCO-2016-01.pdf`.

[5] 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.

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

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

[8] V. Breschi, A. Bemporad, and D. Piga. Identification of hybrid and linear parameter varying models via recursive piecewise affine regression and discrimination. In *European Control Conference*, Aalborg, Denmark, 2016.

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

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

[11] 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.

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

[13] 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.

[14] 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.

[15] F. Lauer. On the complexity of piecewise affine system identification. *Automatica*, 62:148–153, 2015.

[16] 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`.

[17] 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.

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

[19] H. Ohlsson and L. Ljung. Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, 49(4):1045–1050, 2013.

[20] 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.

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

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

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

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