Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 54-7 (2021) 421–426

Piecewise nonlinear regression with data augmentation

M. Mazzoleni *, **V. Breschi** **, **S. Formentin** **

* *Department of Management, Information and Production engineering, University of Bergamo, via Marconi 5, 24044 Dalmine (BG), Italy (e-mail: mirko.mazzoleni@unibg.it).*

** *Department of Electronics, Information and Bioengineering, Politecnico di Milano, via G. Ponzio 34/5, 20133 Milano, Italy (e-mail: {valentina.breschi,simone.formentin}@polimi.it)*

Abstract: Piecewise regression represents a powerful tool to derive accurate yet modular models describing complex phenomena or physical systems. This paper presents an approach for learning PieceWise NonLinear (PWNL) functions in both a supervised and semi-supervised setting. We further equip the proposed technique with a method for the automatic generation of additional unsupervised data, which are leveraged to improve the overall accuracy of the estimate. The performance of the proposed approach is preliminarily assessed on two simple simulation examples, where we show the benefits of using nonlinear local models and artificially generated unsupervised data.

Keywords: Hybrid System Identification; Nonparametric Methods; Nonlinear System Identification

1. INTRODUCTION

Piecewise models can be used to describe highly and possibly discontinuous nonlinear physical systems and phenomena, while being fairly interpretable. Indeed, they allow to approximate complex relationships between a target output $y \in \mathbb{R}$ and a feature vector $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ with a finite collection of local models, defined over a partition of the feature space $\mathcal{X}$. Within the control community, most of research efforts have been devoted to devise procedures to learn *PieceWise Affine (PWA) models* from data (see Garulli et al. (2012) for an overview), since these models can be analyzed with off-the-shelf tools and straightforwardly used for control design, *e.g.,* see Bemporad and Morari (1999). This learning problem is known to be NP-hard, Lauer (2015), despite the simple structure of the local models. Several approaches have been proposed in the last decades to handle PWA regression, *e.g.,* Roll et al. (2004); Juloski et al. (2005); Ohlsson and Ljung (2013); Breschi et al. (2016).

Despite their appealing features, namely the linear structure of the sub-models and the existence of effective learning tools, PWA models might become overly complex if the system/phenomenon to be described is highly nonlinear within the same operating region. In this case, one might need more than one sub-model to describe the behavior of the underlying system/phenomena at a given working condition, with a price paid in terms of model interpretability. To avoid this problem, we *embed kernel regression* Pillonetto et al. (2014) into the approach proposed in Bemporad et al. (2018), so as to *learn PieceWise NonLinear (PWNL) models* in a nonparametric fashion.

Few attempts have already been made in this direction. In Lauer and Bloch (2008), PWNL regression is solved through the combination of kernel regression and support vector machines (SVMs). Since *all* data points are exploited to construct each local kernel, the number of optimization variables critically increases with the dimension of the dataset, thus limiting the applicability of the approach. This computational issue is overcome in Lauer et al. (2010), where heuristics are introduced to reduce the number of parameters to be learned *before* starting the optimization. In Lauer and Bloch (2014), piecewise smooth systems are identified by solving an optimization problem that trades-off between data fitting and model complexity. The outcome is then exploited to find the partition of the feature space through a separate classification task. Instead, our approach embeds the three learning tasks (regression, clustering and classification) within the same optimization routine.

Existing works on PWNL regression are all tailored for fully supervised settings only. Nonetheless, in practical scenarios, a set of features might be available without the corresponding output. Instead of discarding additional data, we equip the proposed PWNL regression technique with a method to exploit *unsupervised regressors*, that are used to improve the reconstructed polyhedral partition of the feature space. The obtained semi-supervised piecewise nonlinear regression method, called GREEDY-SS PWNL, is also equipped with the automatic generation of *unsupervised data points*, by tailoring the works of Formentin et al. (2019); Mazzoleni et al. (2018); Mazzoleni et al. (2018), so as to benefit from additional features when not provided.

The contributions of the paper can be summarized as follows. We (*i*) present a novel method for PWNL regression inspired by Bemporad et al. (2018); (*ii*) we extend this approach to a semi-supervised setting; (*iii*) we propose a data augmentation strategy to exploit the benefits of the

10.1016/j.ifacol.2021.08.396

semi-supervised framework even when unsupervised data are not originally provided.

2. SETTING AND GOAL

Assume that the relationship between a feature vector $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ and the corresponding output $y \in \mathbb{R}$ is described by an unknown nonlinear and possibly discontinuous function $f_{\text{o}} : \mathcal{X} \to \mathbb{R}$, namely $y = f_{\text{o}}(\boldsymbol{x})$. We aim at approximating the target mapping f_{o} with a finite collection of nonlinear maps, defined over a complete polyhedral partition [1] of the feature space $\mathcal{X}$. We learn a *piecewise nonlinear* (PWNL) map $f : \mathcal{X} \to \mathbb{R}$, defined as

$$f(x) = \begin{cases} g_1(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in \mathcal{X}_1, \\ \vdots & \\ g_M(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in \mathcal{X}_M, \end{cases} \tag{1}$$

where M is number of the of *unknown* and possibly nonlinear functions $g_m : \mathcal{X}_m \to \mathbb{R}$ shaping each local approximator, and $\mathcal{X}_m \in \mathcal{X}$ is the m-th polyhedron [2] characterizing the partition of space $\mathcal{X}$, with $m = 1, \dots, M$. Throughout the rest of the paper, we assume that each unknown local model g_m belongs to a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_m$ associated with a kernel $k_m : \mathcal{X}_m \times \mathcal{X}_m \to \mathbb{R}$, see Aronszajn (1950), implying that the local models might not share the same structure. To accomplish our learning task, we consider two different scenarios. Initially, we assume that a dataset $\mathcal{D}_{N_s} = \{\boldsymbol{x}_n, y_n\}_{n=1}^{N_s}$ is available to learn f in (1), with the measured outcomes $\{y_n\}_{n=1}^{N}$ corrupted by a zero-mean, additive random noise independent from $\boldsymbol{x}_n$, for all $n = 1, \dots, N_s$. Then, we assume that an additional set of unsupervised features is available to estimate the PWNL approximating function f. Independently of the considered setting, learning a PWNL approximating function f from experimental data amounts to: (i) select the number of local models M; (ii) estimate the M nonlinear sub-models and (iii) find the polyhedral partition $\{\mathcal{X}_m\}_{m=1}^{M}$ of the feature domain $\mathcal{X}$. In this work, M is assumed to be fixed by the user.

3. SUPERVISED PWNL REGRESSION

To identify a PWNL map from data we exploit an instance of the approach presented in Bemporad et al. (2018), tailored to handle nonlinear sub-models and to provide an explicit polyhedral partition of the feature space.

Let $\boldsymbol{X}_s = \{\boldsymbol{x}_n\}_{n=1}^{N_s}$ and $Y = \{y_n\}_{n=1}^{N_s}$ be the available collections of features and outputs, respectively. Denote with $s_n \in \{1, \dots, M\}$ the *latent variable* indicating the active sub-model for the n-th data point, and let $\mathcal{S}$ be the sequence of active modes, *i.e.*, $\mathcal{S} = \{s_n\}_{n=1}^{N_s}$. Even though this sequence is not explicitly mentioned among the unknowns of our model in Section 2, the estimation of $\mathcal{S}$ enables us to cluster the available data points and to find both the local models and the polyhedral partition of $\mathcal{X}$ within a single optimization routine. According to the chosen framework, the problem of learning a PWNL approximating function from $\mathcal{D}_{N_s}$ can be formalized as the following optimization problem:

$$\min_{\mathcal{G}_M, \boldsymbol{\Theta}_x, \mathcal{S}} \quad \ell_f(Y, \boldsymbol{X}_s, \mathcal{G}_M, \mathcal{S}) + \ell_p(\boldsymbol{X}_s, \boldsymbol{\Theta}_x, \mathcal{S}), \tag{2}$$

where $\mathcal{G}_M = \{g_m \in \mathcal{H}_m\}_{m=1}^{M}$ is the collection of local *unknown* sub-models and $\boldsymbol{\Theta}_x$ are the parameters characterizing the polyhedral partition of $\mathcal{X}$. The first term in the cost $\ell_f(Y, \boldsymbol{X}_s, \mathcal{G}_M, \mathcal{S})$ penalizes *local models fitting errors* and it is defined as follows:

$$\ell_f(Y, \boldsymbol{X}_s, \mathcal{G}_M, \mathcal{S}) = \sum_{n=1}^{N_s} \left(y_n - g_{s_n}(\boldsymbol{x}_n)\right)^2 + \lambda_T \sum_{m=1}^{M} \|g_m\|_{\mathcal{H}_m}^2, \tag{3}$$

where $\|g_m\|_{\mathcal{H}_m}^2$ is a Tikhonov regularization term and the parameter $\lambda_T \in \mathbb{R}^+$ controls the regularization strength. Instead, the second term $\ell_p(\boldsymbol{X}_s, \boldsymbol{\Theta}_x, \mathcal{S})$ quantifies the accuracy of the polyhedral partition characterized by the parameters $\boldsymbol{\Theta}_x = (\boldsymbol{\theta}_{x,1}, \dots, \boldsymbol{\theta}_{x,M})$. To obtain a closed-form expression for such a cost, we search for a PWA separator of the clusters dictated by $\mathcal{S}$. Namely, we seek

$$\phi(\boldsymbol{x}) = \max_{m=1\dots M} \{\phi_m(\boldsymbol{x})\}, \quad \text{with } \phi_m(\boldsymbol{x}) = \boldsymbol{\theta}_{x,m}^\top \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}, \tag{4}$$

such that the following holds:

$$\begin{cases} \phi(\boldsymbol{x}) = \boldsymbol{\theta}_{x,m}^\top \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}, & \forall \boldsymbol{x} \in \mathcal{X}_m, m = 1, \dots, M \\ \phi(\boldsymbol{x}) \geq \boldsymbol{\theta}_{x,j}^\top \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} + 1, & \forall \boldsymbol{x} \in \mathcal{X}_m, m \neq j. \end{cases} \tag{5}$$

We stress that, by definition, each affine function ϕ_m now dictates the boundaries of a polyhedron in $\mathcal{X}_m$, namely

$$\mathcal{X}_m = \{\boldsymbol{x} \in \mathbb{R}^{n_x} : \phi(\boldsymbol{x}) = \phi_m(\boldsymbol{x})\}, \quad m = 1, \dots, M. \tag{6}$$

To this end, we exploit the same rationale of Breschi et al. (2016) and we select the *partition cost* $\ell_p(\boldsymbol{X}_s, \boldsymbol{\Theta}_x, \mathcal{S})$ as:

$$\ell_p(\boldsymbol{X}_s, \boldsymbol{\Theta}_x, \mathcal{S}) = \sum_{n=1}^{N_s} \ell_x(\boldsymbol{x}_n, \boldsymbol{\Theta}_x, s_n) + \lambda_{T_p} \|\boldsymbol{\Theta}_x\|_2^2, \tag{7a}$$

$$\ell_x(\boldsymbol{x}_n, \boldsymbol{\Theta}_x, s_n) = \sum_{\substack{j=1 \\ j \neq s_n}}^{M} \max\left\{0, (\boldsymbol{\theta}_{x,j} - \boldsymbol{\theta}_{x,s_n})^\top \begin{bmatrix} \boldsymbol{x}_n \\ 1 \end{bmatrix} + 1\right\}^2, \tag{7b}$$

so to weight the average violation of the conditions in (5), with $\lambda_{T_p} \in \mathbb{R}^+$ dictating the strength of the $\mathcal{L}_2$-regularization.

As summarized in Algorithm 1, Problem (2) is solved by alternatively learning $\{g_m\}_{m=1}^{M}$ and $\boldsymbol{\Theta}_x = (\boldsymbol{\theta}_{x,1}, \dots, \boldsymbol{\theta}_{x,M})$, for a fixed sequence $\mathcal{S}$, and updating $\mathcal{S}$ with the sub-models and the partition fixed. In the proposed learning procedure, we exploit an additional feature of the considered problem, namely the separability of (2) with respect to $\{g_m\}_{m=1}^{M}$ and $\boldsymbol{\Theta}_x$ for a given $\mathcal{S}$. This allows us to employ the computationally efficient multi-category discrimination technique proposed in Breschi et al. (2016) to estimate the PWA separator (see step 1.2).

3.1 Nonparametric local model estimation

Consider the problem of estimating the local models from data, at the i-th iteration, for a fixed sequence $\mathcal{S}^{i-1}$ at step 1.1 of Algorithm 1, namely

$$\min_{\mathcal{G}_M} \quad \sum_{n=1}^{N_s} \left(y_n - g_{s_n^{i-1}}(\boldsymbol{x}_n)\right)^2 + \lambda_T \cdot \sum_{m=1}^{M} \|g_m\|_{\mathcal{H}_m}^2. \tag{8a}$$

Let $\mathcal{X}_m^i$ be the set of points associated to the m-th mode at the i-th iteration, and $\mathcal{N}_m^i = \{n \in [1,\ N_s] : s_n^{i-1} =$

[1] A complete polyhedral partition $\{\mathcal{X}_m\}_{m=1}^{M}$ verifies: $\bigcup_{m=1}^{M} \mathcal{X}_m = \mathcal{X}$ and $\mathring{\mathcal{X}}_i \cap \mathring{\mathcal{X}}_j = \emptyset\ \forall i \neq j$, $i, j = 1, \dots, M$, where $\mathring{\mathcal{X}}_i$ is the interior of $\mathcal{X}_i$.

[2] A polyhedron $\mathcal{X}_m$ is described by a set of real-valued linear inequalities.

Algorithm 1 [Supervised PWNL regression]

Input: Data $\{Y, \boldsymbol{X}_s\}$; initial sequence $\mathcal{S}^0$; $\lambda_T, \lambda_{T_p} \in \mathbb{R}^+$.

1. **for** $i = 1, \dots$ **do**
   1.1. $\mathcal{G}_M^i \leftarrow \arg\min_{\mathcal{G}_M} \ell_f(Y, \boldsymbol{X}_s, \mathcal{G}_M, \mathcal{S}^{i-1})$
   1.2. $\boldsymbol{\Theta}_x^i \leftarrow \arg\min_{\boldsymbol{\Theta}_x} \ell_p(\boldsymbol{X}_s, \boldsymbol{\Theta}_x, \mathcal{S}^{i-1})$
   1.3. $\mathcal{S}^i \leftarrow \arg\min_{\mathcal{S}} \sum_{n=1}^{N_s} \ell\left(y_n, \boldsymbol{x}_n, \mathcal{G}_M^i, \boldsymbol{\Theta}_x^i, s_n\right)$
2. **until** $\mathcal{S}^i = \mathcal{S}^{i-1}$

Output: Local models $\mathcal{G}_M^\star = \mathcal{G}_M^i$; separator $\boldsymbol{\Theta}_x^\star = \boldsymbol{\Theta}_x^i$.

$m\}$ the set of their indexes with $|\mathcal{N}_m^i|$ its cardinality. By straightforward manipulations it can be shown that minimizing the cost in (8a) corresponds to solving M distinct optimization problems

$$\min_{g_m} \sum_{n \in \mathcal{N}_m^i} \left(y_n - g_m\left(\boldsymbol{x}_n\right)\right)^2 + \lambda_T \cdot \|g_m\|_{\mathcal{H}_m}^2, \tag{8b}$$

with $m = 1, \dots, M$. Therefore, step 1.1 involves the solution of M separate nonlinear regression sub-problems, each one leading to an updated estimate of a sub-model.

According to the *representer theorem*, see Dinuzzo and Schölkopf (2012), problem (8b) admits the solution

$$g_m(\boldsymbol{x}) = [\boldsymbol{\theta}_{y,m}^i]^\top \cdot \mathbf{h}, \tag{9}$$

where $\boldsymbol{\theta}_{y,m}^i$ is a vector of coefficients, and the vector $\mathbf{h} \in \mathbb{R}^{|\mathcal{N}_m^i|}$ stacks the values $k_m(\boldsymbol{x}, \boldsymbol{x}_r)$, where $r \in \mathcal{N}_m^i$. Thus, problem (8b) can be restated using (9) as

$$\min_{\boldsymbol{\theta}_{y,m}^i} \|\boldsymbol{y}_m^i - \mathbf{K}_m^i \boldsymbol{\theta}_{y,m}^i\|_2^2 + \lambda_T \cdot (\boldsymbol{\theta}_{y,m}^i)^\top \mathbf{K}_m^i \boldsymbol{\theta}_{y,m}^i, \tag{10}$$

where $\boldsymbol{y}_m^i \in \mathbb{R}^{|\mathcal{N}_m^i|}$ is the vector stacking the outputs associated with the m-th mode, and $\mathbf{K}_m^i \in \mathbb{R}^{|\mathcal{N}_m^i| \times |\mathcal{N}_m^i|}$ is a semidefinite positive, symmetric matrix with elements $k_m(\boldsymbol{x}_r, \boldsymbol{x}_c)$, with $\boldsymbol{x}_r, \boldsymbol{x}_c \in \mathcal{X}_m^i$. As such, these updates boil down to M kernel ridge regression problems Saunders et al. (1998).

Problem (10) admits the closed form solution

$$\boldsymbol{\theta}_{y,m}^i = \left[\mathbf{K}_m^i + \lambda_T \cdot I_{|\mathcal{N}_m^i|}\right]^{-1} \boldsymbol{y}_m^i. \tag{11a}$$

with $I_{|\mathcal{N}_m^i|}$ the identity matrix. Since $\mathcal{N}_m^i$ is constructed according to the fixed sequence $\mathcal{S}^{i-1}$, we stress that the parameters vector $\boldsymbol{\theta}_{y,m}^i \in \mathbb{R}^{|\mathcal{N}_m^i|}$ changes its dimension *iteratively* along with the kernel matrix $\mathbf{K}_m^i$.

Remark 1. (Computational hints). At each iteration we use only a subset of data to build the matrices $\{\mathbf{K}_m^i\}_{m=1}^M$, due to their dependency on the updated mode sequence. An efficient way to construct $\mathbf{K}_m^i$ is to first compute the full matrix (considering all N_s points), and then index it with only the ones in the set $\mathcal{N}_m^i$. ■

3.2 Mode sequence update

By looking at the objective function in (2), it can be noticed that the latter is separable with respect to $\{s_n\}_{n=1}^{N_s}$ for given local models and a fixed partition. Moreover, there is clearly no dependence between consecutive values of the latent variable within $\mathcal{S}$. This allows us to refine the mode sequence iteratively in a rather straightforward way, while accounting for both *local model fitting* and the quality of the partition estimated based on the previous estimate of the mode sequence.

Based on the chosen cost function, let the cost of assigning a pair $\{\boldsymbol{x}_n, y_n\}$ to the m-th mode at the i-th iteration be

$$L(m, n) = \ell(\boldsymbol{x}_n, y_n, \mathcal{G}_m^i, \boldsymbol{\Theta}_x^i, m), \tag{12a}$$

with

$$\ell\left(\boldsymbol{x}_n, y_n, \mathcal{G}_M^i, \boldsymbol{\Theta}_x^i, m\right) = \left(y_n - g_m^i\left(\boldsymbol{x}_n\right)\right)^2 + \\ + \sum_{\substack{j=1 \\ j \neq m}}^{M} \max\left\{0, \left(\boldsymbol{\theta}_{x,j}^i - \boldsymbol{\theta}_{x,m}^i\right)^\top \left[\begin{smallmatrix} \boldsymbol{x}_n \\ 1 \end{smallmatrix}\right] + 1\right\}^2, \tag{12b}$$

which can be computed exploiting the updated estimates $\{g_m^i\}_{m=1}^M$ and the parameters $\boldsymbol{\Theta}_x^i$. This *allocation cost* in (12b) is independent from the regularization terms, since they do not depend on the mode sequence.

By relying on the characteristics of the considered problem and the allocation cost defined in (12), the mode associated to each data point is reconstructed as

$$s_n^i = \arg\min_{m=1,\dots,M} L(m, n), \quad n = 1, \dots, N_s. \tag{13}$$

Remark 2. (Jump models). Algorithm 1 can be extended to nonlinear jump models by reconstructing the mode sequence via *dynamic programming* (DP) Bertsekas (2000). This generalization is the subject of ongoing research. ■

Remark 3. (Validation hints). To predict the output associated with a *fresh* set of features $(\boldsymbol{x}_1^*, \dots, \boldsymbol{x}_{N_{\text{val}}}^*)$, one has to (i) assign the new sample to one of the modes according to the estimated polyhedral partition, and then (ii) reconstruct the output through the estimated model associated to the chosen mode. ■

4. GREEDY SEMI-SUPERVISED PWNL (GREEDY-SS PWNL) REGRESSION

Suppose that, in addition to the supervised dataset $\mathcal{D}_{N_s}$, an unsupervised set $\mathcal{D}_{N_u}$ is also accessible, which includes N_u feature vectors $\boldsymbol{x}_{n,u}$ for which the target output is not available. Since the approximating map f in (1) is defined over a polyhedral partition of the feature space $\mathcal{X}$, these additional data can be exploited quite straightforwardly within our framework to estimate the polyhedral partition.

Let $\boldsymbol{X}$ be the collection of available $N = N_s + N_u$ feature vectors, namely $\boldsymbol{X} = \{\tilde{\boldsymbol{x}}_n\}_n^N$ with

$$\tilde{\boldsymbol{x}}_n = \begin{cases} \boldsymbol{x}_n & \text{if} \quad 1 \leq n \leq N_s, \\ \boldsymbol{x}_{n-N_s,u} & \text{if} \quad N_s + 1 < n \leq N. \end{cases} \tag{14}$$

To exploit the unsupervised features in the computation of the polyhedral partition, the cost in (2) can be slightly modified as follows:

$$\min_{\mathcal{G}_M, \boldsymbol{\Theta}_x, \tilde{\mathcal{S}}} \quad \ell_f(Y, \boldsymbol{X}_s, \mathcal{G}_M, \tilde{\mathcal{S}}) + \ell_p(\boldsymbol{X}, \boldsymbol{\Theta}_x, \tilde{\mathcal{S}}), \tag{15}$$

with the partition cost $\ell_p(\boldsymbol{X}, \boldsymbol{\Theta}_x, \tilde{\mathcal{S}})$ still defined as in (7), but now depending on both supervised and unsupervised features. This implies that the new optimization variable $\tilde{\mathcal{S}}$ collects the modes associated to both supervised and unsupervised samples, *i.e.*, $\tilde{\mathcal{S}} = \{\tilde{s}_n\}_{n=1}^N$.

Since the cost in (15) is still separable with respect to $\mathcal{G}_M$ and $\boldsymbol{\Theta}_x$, the local models and the partition can still be learned separately. Moreover, as the fitting cost

$\ell_f(Y, \boldsymbol{X}_s, \mathcal{G}_M, \tilde{\mathcal{S}})$ is independent from unsupervised data, the local models are still estimated exactly as in step 1.1 of Algorithm 1. The partition is found by computing a PWA separator through the Newton-like method proposed in Breschi et al. (2016), now relying on the additional features available. As the polyhedral partition is shaped by unsupervised data, they directly affect the reconstructed mode sequence and indirectly influence the learned local models. The mode sequence is still found as discussed in Section 3.2, by considering a different allocation cost with respect to the one in (12b). In particular, the new cost associated to the m-th mode is given by $\ell\left(\tilde{\boldsymbol{x}}_n, y_n, \mathcal{G}_M^i, \boldsymbol{\Theta}_x^i, m\right)$ in (12b) for supervised points, thus accounting for both local fitting and misclassification, while it is equal to $\ell_x(\tilde{\boldsymbol{x}}_n, \boldsymbol{\Theta}_x^i, m)$ in (7b) for unsupervised features, thus depending on their position within $\mathcal{X}$ only.

5. AN APPROACH TO DATA AUGMENTATION

When a set of unsupervised features $\mathcal{D}_{N_u}$ is not available, one can augment the data by creating this set *artificially*. Following the same rationale in Formentin et al. (2019), we propose a simple, yet effective, approach for the automatic generation of additional features, by exploiting the hallmarks of the considered model.

5.1 Automatic unsupervised data generation

Let the modes of the supervised data be fixed and denote as $\mathcal{N}_{m,s} = \{n \in [1,\ N_s] : s_n = m\}$ the set of indexes of supervised data points associated with the m-th mode. For each sample $\boldsymbol{x}_n$ such that $n \in \mathcal{N}_{m,s}$, we generate η unsupervised data points as

$$\boldsymbol{x}_{n,u}^j = \boldsymbol{x}_n + \boldsymbol{v}_n^j, \quad j = 1, \dots, \eta, \tag{16}$$

where $\boldsymbol{v}_n^j$ is a random vector and $\eta > 0$ is a parameter to be tuned so as to balance the improvement attained by augmenting the dataset and overfitting the available data. The value of $\boldsymbol{v}_n^j$ determines the distance of the unsupervised data point $\boldsymbol{x}_n^j$ to the supervised one $\boldsymbol{x}_n$. On the one hand, this has to be sufficiently large so to exhaustively explore the feature space $\mathcal{X}$. On the other hand, $\boldsymbol{v}_n^j$ should be small enough to guarantee that the newly generated points lie in the same region of the supervised features. To satisfy these constraints, one can impose $\boldsymbol{v}_n^j$ to be uniformly distributed [3], namely

$$\boldsymbol{v}_n^j \sim U_{[-h_n, h_n]}, \quad j = 1, \dots, \eta, \tag{17}$$

where $h_n > 0$ are parameters that determine the area where unsupervised points are created. These parameters can be selected automatically accounting for both our conflicting objectives. Indeed, to create high density clusters within each polyhedral region, we can impose that

$$\|\boldsymbol{x}_{n,u}^j - \boldsymbol{x}_n\| \leq \frac{d_e^m}{2}, \quad j = 1, \dots, \eta, \tag{18}$$

with d_e^m denoting the Euclidean distance between the two closest supervised feature vectors within the m-th cluster. At the same time, for the unsupervised points to be far from the boundaries of the polyhedral regions, we enforce

$$\|\boldsymbol{x}_{n,u}^j - \boldsymbol{x}_n\| \leq \frac{d_{b,n} - \varepsilon}{2}, \quad j = 1, \dots, \eta, \tag{19}$$

[3] Alternative distributions with limited support can also be chosen.

where $d_{b,n}$ is the minimum distance between the n-th supervised sample and the separating hyperplanes characterizing the partition of $\mathcal{X}$, i.e.

$$d_{b,n} = \min_{i=1,\dots,M} \frac{\left|\boldsymbol{\theta}_{x,i}^\top \begin{bmatrix} \boldsymbol{x}_n \\ 1 \end{bmatrix}\right|}{\|\boldsymbol{\theta}_{x,i}^\top \begin{bmatrix} \mathbf{1}_{n_x} \\ 0 \end{bmatrix}\|_2}. \tag{20}$$

The additional parameter $\varepsilon \in [0,\ \max_n [d_{b,n}])$, is introduced as a *safety guard* to guarantee that the distance between the border of the polyhedral regions and the generated point is at least ε. Let $d_n^\star = \min\{d_{b,n} - \varepsilon, d_e^m\}$. According to Formentin et al. (2019), we can set the boundaries of the uniform distribution on $\boldsymbol{v}_n^j$ as

$$h_n = \frac{d_n^\star}{2\sqrt{n_x}}; \quad \frac{d_n^\star}{2} = \min\left\{\frac{d_{b,n}^m - \varepsilon}{2}, \frac{d_e^m}{2\alpha}\right\}, \tag{21}$$

where $\alpha > 0$ is introduced to control the width of the area where the unsupervised points are created. Fig. 1 depicts two possible scenarios of unsupervised data creation. On the left, all supervised points are far from the boundaries of the polyhedral regions, so that the unsupervised points are created according to intra-cluster distances only. In the right plot, one of the available samples is close to the boundaries of the region and, thus, the associated unsupervised points are created according to the distance from the separating hyperplane.

5.2 Embed automatic unsupervised data generation into GREEDY-SS PWNL

The scheme for artificial data generation can be embedded within the semi-supervised PWNL regression method introduced in Section 4, as summarized in Algorithm 2.

Let $\mathcal{S}^{i-1}$ be the active mode sequence associated to the supervised data, computed at the $i-1$-th run of Algorithm 2. At the i-th iteration this sequence is used to update the local models as discussed in Section 3.1, as in Algorithm 1. This step is still independent on unsupervised points. Then, new artificial features are created according to the current PWA separator and the mode sequence $\mathcal{S}^{i-1}$. Because of the dependence of $\{h_n\}_{n=1}^{N_s}$ in (21) on the polyhedral partition, we stress that unsupervised data points have necessarily to be discarded at the end of each run of Algorithm 2 and they must be re-generated at every new iteration. To exploit the additional features in the computation of the PWA separator, the associated mode have also to be initialized. To this end, we rely on $\mathcal{S}^{i-1}$. Specifically, artificial samples are labeled according to the mode associated to the supervised points from which they are generated, namely

$$s_{n,u}^j = s_n^{i-1}, \quad j = 1, \dots, \eta, \tag{22}$$

with $s_{n,u}^j$ being the mode associated to the j-th artificial features generated from the n-th supervised data point. This allows us to construct the extended mode sequence $\tilde{\mathcal{S}}^{i-1}$. At the end of each iteration (see step 1.7 of Algorithm 2), only the subsequence of $\mathcal{S}$ associated to the supervised points is retained, while the last N_u estimated modes are discarded.

Remark 4. (Initialization hints). Since the PWA separator is not initialized, one can generate the unsupervised points by looking at intra-cluster distances only, or run Algorithm 1 and use the estimated partition for the creation of unsupervised points at the first run of Algorithm 2. ■

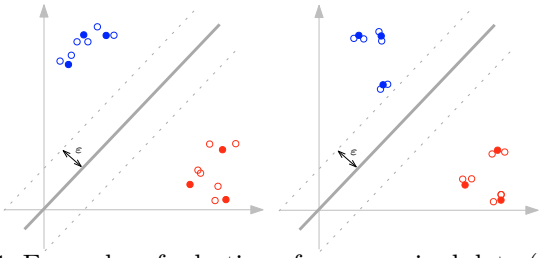

Fig. 1. Examples of selection of unsupervised data (empty circles) with two possible configurations of supervised features (full circles), with $\eta = 2$.

Algorithm 2 [Greedy semi-supervised PWNL regression with artificial feature generation]

Input: Data $\{\boldsymbol{X}_s, Y\}_{n=1}^{N_s}$; initial sequence $\mathcal{S}^0$; $\lambda_T, \lambda_{T_p} \in \mathbb{R}^+$, initial separators parameters $\boldsymbol{\Theta}_x^0$

1. **for** $i = 1, \dots$ **do**
   1.1. $\mathcal{G}_M^i \leftarrow \arg\min_{\mathcal{G}_M} \ell_f(Y, \boldsymbol{X}_s, \mathcal{G}_M, \mathcal{S}^{i-1})$
   1.2. Generate the artificial features
   1.3. Construct $\boldsymbol{X}$ as in (14)
   1.4. Initialize the extended mode sequence $\tilde{\mathcal{S}}^{i-1}$
   1.5. $\boldsymbol{\Theta}_x^i \leftarrow \arg\min_{\boldsymbol{\Theta}_x} \ell_p(\boldsymbol{X}, \boldsymbol{\Theta}_x, \tilde{\mathcal{S}}^{i-1})$
   1.6. Find $\tilde{\mathcal{S}}^i$ as

$$\min_{\tilde{\mathcal{S}}} \sum_{n=1}^{N_s} \ell(\tilde{\boldsymbol{x}}_n, y_n, \mathcal{G}_M^i, \boldsymbol{\Theta}_x^i, \tilde{s}_n) + \sum_{n=N_s+1}^{N} \ell_x(\tilde{\boldsymbol{x}}_n, \boldsymbol{\Theta}_x^i, \tilde{s}_n)$$

   1.7. $\mathcal{S}^i \leftarrow \{\tilde{s}_n^i\}_{i=1}^{N_s}$
2. **until** $\mathcal{S}^i = \mathcal{S}^{i-1}$

Output: Local models $\mathcal{G}_M^\star = \mathcal{G}_M^i$; separator $\boldsymbol{\Theta}_x^\star = \boldsymbol{\Theta}_x^i$.

6. SIMULATION RESULTS

The effectiveness of the proposed approaches is assessed on two simple simulation examples. In both cases, the regularization hyper-parameters λ, λ_{T_p} are fixed in advance and shared by all methods, along with the hyper-parameter σ_k characterizing the Gaussian kernel

$$k_m(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{\sigma_k}\right)}$$

exploited to solve the M nonlinear regression subproblems. We leave the discussion on possible tuning strategies of these important parameters to future work. The performance of the approaches is assessed on a validation set, by looking at the Best Fit Rate (BFR):

$$\text{BFR} = \max\left\{1 - \frac{\|\boldsymbol{y}_{\text{val}} - \hat{\boldsymbol{y}}_{\text{val}}\|}{\|\boldsymbol{y}_{\text{val}} - \bar{y}_{\text{val}}\|}, 0\right\} \cdot 100,$$

where $\boldsymbol{y}_{\text{val}}$ stacks the measured validation outputs, $\hat{\boldsymbol{y}}_{\text{val}}$ collects the ones reconstructed with the estimated model and $\bar{y}_{\text{val}}$ is the sample mean of the measured outputs. Throughout training and validation, the true class associated to each data-point is assumed to be unknown and the actual mode sequence is only used as a ground-truth for performance assessment.

The considered case studies have been chosen so as to show the advantages of using nonlinear sub-models, while highlighting the limitations of the PWA rationale. To this end, we compare the results attained with the proposed PWNL regression methods with the ones obtained through PWA regression. For the comparison to be fair, the PWA models

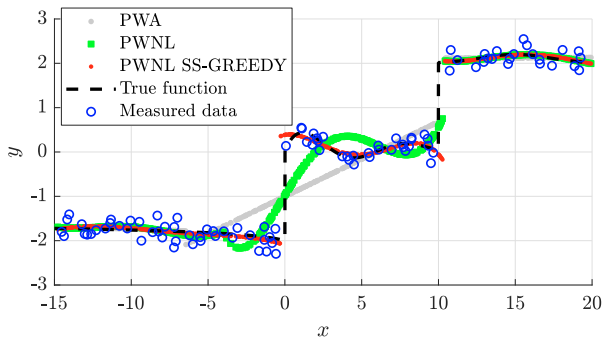

Fig. 2. Example 1. True VS reconstructed functions.

Table 1. Example 1. Validation performance.

	PWA	PWNL	GREEDY-SS PWNL
BFR [0 - 100]	70.84	77.63	**78.53**
Misclassified [%]	16%	13%	**2%**

are retrieved via the approach proposed in Bemporad et al. (2018) by minimizing the loss in (15), properly modified to account for the simpler nature of the local models.

6.1 Example 1. Single-variable function

We randomly generate $N_s = 100$ supervised data from the map

$$f_o(x) = \begin{cases} -2 + 0.1\log(1-x), & \text{if } x \le 0, \\ 1 + 0.01x + \sin(x)/x, & \text{if } 0 < x \le 10, \\ 2 + 0.2\exp\left(-\frac{1}{2}\frac{(x-15)^2}{5}\right), & \text{if } 10 < x \le 20, \end{cases}$$

with the measured output y corrupted by additive noise, such that the Signal-Noise-Ratio (SNR) is 10. The SNR is computed as the ratio of the noiseless output variance over the noise variance. Algorithm 2 is run with $\alpha = 10^{-2}$ and $\eta = 1$, while the other hyper-parameters are set to $\lambda_T = 5{\cdot}10^{-8}, \lambda_{T_p} = 10^{-1}, \sigma_k = 10^3$. Under the assumption that the number of true modes is known, all methods are run by fixing $M = 3$. Fig. 2 shows the results obtained with the three approaches on an evaluation dataset with $N_{\text{val}} = 500$ points. For the given number of modes, both PWNL models are more accurate than the PWA one, as proved by the attained BFRs reported in Table 1. This was somehow expected, since the underlying function is nonlinear in each input sub-domain. At the same time, the GREEDY-SS PWNL approach, that exploits unsupervised data, better estimates the space partition, as shown by the percentages of misclassified points in Table 1. This implies that the data augmentation rationale leads to better estimates of the sub-domains boundaries which, in turn, translate into better function approximation capabilities.

6.2 Example 2. Multi-variable function

We now consider the static map introduced in Breschi et al. (2016). We thus generate $N_s = 100$ supervised data from

$$f_o(\boldsymbol{x}) = \begin{cases} h(\boldsymbol{x}), & \text{if } -0.5 < h(\boldsymbol{x}) < 0.5, \\ 0.5, & \text{if } h(\boldsymbol{x}) \ge 0.5, \\ -0.5, & \text{if } h(\boldsymbol{x}) \le -0.5, \end{cases} \tag{23}$$

with $h : \mathbb{R}^3 \to \mathbb{R}$, $h(\boldsymbol{x}) = 0.6\sin(x_1 + x_2^2 - x_3)$ and the features $\boldsymbol{x} = [x_1\ x_2\ x_3]^\top \in \mathbb{R}^3$ being generated as a white noise sequence with uniform distribution in $[-1\ 1]^3$. The corresponding output, which is corrupted by noise

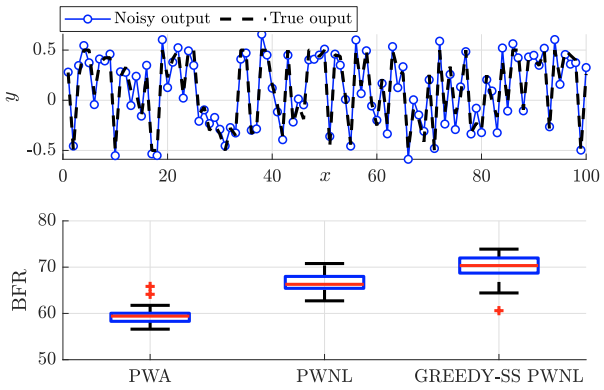

Fig. 3. Example 2. Top: True function VS sampled noisy data. Bottom: Validation results.

Table 2. Example 2. Validation: Median BFRs.

	PWA	PWNL	GREEDY-SS PWNL
BFR [0 - 100]	59.41	66.29	**70.34**

Table 3. Example 2. Fitting: Median CPU time.

	PWA	PWNL	GREEDY-SS PWNL
Time [s]	**0.0086**	0.0088	0.0406

(SNR = 5), is shown in Fig. 3. The performance of the different models is assessed through 100 Monte Carlo simulations varying the realization of the noise on training data. Algorithm 2 is run with $\alpha = 10^{-2}$ and $\eta = 3$, while the other hyper-parameters are set to $\lambda_T = 10^{-5}, \lambda_{T_p} = 1, \sigma_k = 10^3$. The models are estimated by fixing $M = 3$. The results on $N_{\text{val}} = 500$ unseen points are shown in Fig. 3, while Table 2 reports the median performance of the three algorithms. The model obtained with GREEDY-SS PWNL outperforms the other two, leading to a reduced reconstruction error while being robust to the different realizations of the training set. Table 3 reports the median CPU time[4] required by the methods, across all the 100 simulations. The time required to fit the PWNL model is similar to the one needed to learn the PWA model, thus proving the computational efficiency of the proposed approach despite the introduction of kernels. As expected, the GREEDY-SS PWNL approach is the most demanding one, because of the generation of artificial samples.

7. CONCLUSIONS

We presented a nonlinear extension of the approach proposed in Bemporad et al. (2018) for PWNL regression, which combines kernel methods and multi-category discrimination. Following Formentin et al. (2019), we equipped the proposed learning technique with the automatic generation of unsupervised regressors, to enhance the performance when reconstructing the operating regions of the identified PWNL model. The performance attained on two examples show the benefits of employing the PWNL and GREEDY-SS PWNL strategies over comparable PWA regression techniques when the underlying data-generating system is highly complex and the number of fixed modes is relatively low. Future research will be devoted to the automatic tuning of the hyper-parameters and the comparison with other PWNL approaches.

[4] CPU times were computed on an Intel Xeon E-2176M processor with 32 GB of RAM running MatLab .

REFERENCES

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3), 337–404.

Bemporad, A., Breschi, V., Piga, D., and Boyd, S. (2018). Fitting jump models. *Automatica*, 96, 11 – 21.

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3), 407 – 427.

Bertsekas, D. (2000). *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition.

Breschi, V., Piga, D., and Bemporad, A. (2016). Piecewise affine regression via recursive multiple least squares and multicategory discrimination. *Automatica*, 73, 155 – 162.

Dinuzzo, F. and Schölkopf, B. (2012). The representer theorem for hilbert spaces: a necessary and sufficient condition. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25, 189–196. Curran Associates, Inc.

Formentin, S., Mazzoleni, M., Scandella, M., and Previdi, F. (2019). Nonlinear system identification via data augmentation. *Systems & Control Letters*, 128, 56 – 63.

Garulli, A., Paoletti, S., and Vicino, A. (2012). A survey on switched and piecewise affine system identification. *IFAC Proceedings Volumes*, 45(16), 344 – 355. 16th IFAC Symposium on System Identification.

Juloski, A., Weiland, S., and Heemels, W. (2005). A bayesian approach to identification of hybrid systems. *IEEE Transactions on Automatic Control*, 50(10), 1520–1533.

Lauer, F. (2015). On the complexity of piecewise affine system identification. *Automatica*, 62, 148 – 153.

Lauer, F. and Bloch, G. (2008). Switched and piecewise nonlinear hybrid system identification. In *Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control*, HSCC '08, 330–343.

Lauer, F. and Bloch, G. (2014). Piecewise smooth system identification in reproducing kernel hilbert space. In *53rd IEEE Conference on Decision and Control*, 6498–6503.

Lauer, F., Bloch, G., and Vidal, R. (2010). Nonlinear hybrid system identification with kernel models. In *49th IEEE Conference on Decision and Control (CDC)*, 696–701.

Mazzoleni, M., Formentin, S., Scandella, M., and Previdi, F. (2018). Semi-supervised learning of dynamical systems: a preliminary study. *2018 European Control Conference (ECC)*, 2824–2829.

Mazzoleni, M., Scandella, M., Formentin, S., and Previdi, F. (2018). Identification of nonlinear dynamical system with synthetic data: a preliminary investigation. *IFAC-PapersOnLine*, 51(15), 622 – 627. 18th IFAC Symposium on System Identification SYSID 2018.

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

Pillonetto, G., Dinuzzo, F., Chen, T., De Nicolao, G., and Ljung, L. (2014). Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50(3), 657 – 682.

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

Saunders, C., Gammerman, A., and Vovk, V. (1998). Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, 515–521. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.