This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

Fuzzy space partitioning based on hyperplanes defined by eigenvectors for Takagi-Sugeno fuzzy model identification

Dejan Dovžan, Igor Škrjanc
Faculty of Electrical Engineering, University of Ljubljana
Tržaška 25, SI-1000 Ljubljana, Slovenia

***Abstract*—This paper presents a novel method for fuzzy space partitioning and the identification of Takagi-Sugeno fuzzy models. The novelty is in its region-splitting mechanism and membership function definition, which is based on hyperplanes. The proposed algorithm introduces a concept of principal component analysis to define the hyperplanes that split the problem space and uses the distances to these hyperplanes as metrics instead of center-oriented clusters. In contrast with many other methods, the presented method delivers reproducible results and has an easy tuning procedure. The performance is illustrated with analytical examples, benchmark problems from the literature, and real-process data. The obtained results are very promising; however, as with most learning methods, the results depend on the data distribution and input variable selection.**

***Index Terms*—Hierarchical Fuzzy Model, Eigen-Hyperplane, Principle Component Partitioning.**

I. Introduction

Processes, which are nowadays equipped with a number of sensors, produce a huge amount of data. With the help of these data, models of the processes can be generated, which allow us to better understand the behaviour, and can be used for the optimization or control of the process. Usually, data-mining algorithms and data-based model identification algorithms are used to derive the models from the measured process data. Although some of these methods can produce very accurate results, they sometimes contradict the modeling goals in the process and control industry, i.e., instead of producing simple, transparent, and explanatory models, some methods produce models that are large, complex, highly uninterpretable, and nontransparent. This limits the spread and use of such models and methods in the process industry and control in general. In process and control industry, methods that give reproducible results, transparent and explanatory models, and can also produce dynamical models are preferred. The simplest approach for identifying models based on data is linear regression. However, with this method, only linear models can be identified.

To model a nonlinear process, which is usually the case when dealing with real processes, a multi-model approach can be used [1]. The idea is to combine several simple, usually linear, sub-models and use them to describe the global nonlinear behavior of the process. This concept is incorporated in the Takagi-Sugeno (T-S) fuzzy model [2]. Models in the T-S form are very important nonlinear approximators for nonlinear static and dynamic process approximations [1], [3], [4]. This is mainly due to their transparency and linear local models (LM). This makes an easy extension of classical linear control theory to the nonlinear world.

The identification of the T-S model with known structures consists of the identification of model's premise and consequent part parameters. With the premise parameters, the problem space is divided into smaller regions over which the local linear models (consequence part) are defined to approximate the target data. The partitioning of the problem space determines the validity regions and the validity of the local models. The estimation of the local models is then done using least-squares methods. The decisive difference between existing algorithms in the literature is the strategy of problem space partitioning (defining the validity regions and, consequently, the parameters of validity functions). The space partitioning in our approach is based on a hyperplane division, which is close to the concept of the hinging hyperplane model proposed in [5]. The main advantages of this approach are the simplicity and transparency of the structure. The disadvantages are the convergence and initialization problems, which are reported in [6], [7]. The hinging hyperplane model can be obtained using different methods: by improving the basic Breiman's algorithm given in [5], or by the Gauss-Newton algorithm given in [8], by mixed-integer programming as proposed in [9], or using the structure of hierarchical models as proposed in [7] and [10] together with the fuzzy c-regression method (FCRM) proposed in [11], or the fuzzy c-varieties method to identify hinging hyperplane models. One possible solution for identifying the hyperplanes in the data is the evolving principal component clustering with a low run-time complexity presented in [12]. The hinging hyperplane models contain the information of local linear models and the partitioning of the operating domain given by the validity or membership functions, as is the case in projective motion segmentation [13] or in hybrid system identification [14], [15]. In [16], it is shown that a hyperplane-shaped clustering (HPSC) is more effective in Takagi-Sugeno (T-S) fuzzy model identification compared to hypersphere-shaped clustering, because the hyperplane can be directly accommodated into the rule premise so that the number of fuzzy parameters is significantly reduced. The use of hyperplane clustering in a single-pass learning mode, called parsimonious learning machine (PALM), is proposed in [17]. Because of low computational complexity, due to the use of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

hyperplane clustering, it is very fast and enables effective on-line learning, suitable for the fast and rapidly changing natures of data streams. One of very important hyperplane clustering methods is Algebraic Subspace Clustering (ASC) which results in solutions in the form of the factorization or differentiation of polynomials [18], but it has a major drawback in its exponential growth of complexity. Another method of hyperplane clustering that has a solid theoretical background is Spectral Curvature Clustering (SCC) [19]. This method computes a D-fold affinity between all D-tuples of points in the data set. The method has combinatorial complexity and becomes impractical for higher dimensions. The most practical method is K-Hyperplanes method (KH) [20], which assigns a new cluster and corresponding new normal eigenvector defined by principal component analysis of a cluster covariance matrix (PCA). The method is robust to the noise, but it is sensitive to outliers and the results may converge to the local minimum. A modification of KH is the Median K-Flats method (MKF) [21], which uses L_1 norm in the objective function instead of L_2 norm, as proposed in the KH method. This solves the problem of robustness to outliers. At the end, it should be pointed out that any single robust subspace learning method suitable for high relative dimensions, i.e., RANSAC [22] or REAPER [23], can be applied for sequential extraction of the most dominant hyperplane by removing the points lying close to it; also, the learning of the next most dominant hyperplane can be achieved in an iterative way. This iterative hierarchical procedure was implemented in [24]; it is a supervised hierarchical clustering for fuzzy model identification. The high flexibility of the validity functions that is obtained with the fuzzy clustering combined with supervised learning results in an efficient partitioning algorithm, which is independent of initialization and results in a parsimonious fuzzy model. The supervised hierarchical clustering is very efficiently used in evolving a fuzzy-model-based design of experiments, as reported in [25], [26]. An interesting idea regarding the golden section searching method, which is close to the hyperplane searching method and results in a small number of partitions, is presented in [27]. The evolving strategy of possibilistic clustering for the monitoring of cyber-attacks is presented in [28], and a general purpose evolving clustering with different inner matrix norms is shown in [29]. An extensive overview and analysis of hyperplane clustering methods is given in [30]. The main contribution of this paper is the novel partitioning algorithm based on hyperplanes that are defined by eigenvectors. This partitioning is the most appropriate and the easiest way for the further development of Takagi-Sugeno local linear models. The presented approach is, in certain aspects, related to methods such as SUHICLUST [24], [25], LOLIMOT [31], and HILOMOT [32]. LOLIMOT considers splitting the partition into two (or more, depending on user settings) equal parts in each dimension. The split is performed in the dimension that decreases the model error. The disadvantage of this is that the partitioning is axis-parallel. Furthermore, with high dimensional problems, the checking of splits in each dimension is time-consuming. The drawback of axis-parallel partitioning was fixed using the HILOMOT method. This method uses the same principle as LOLIMOT with the additional optimization of a splitting border angle. However, this introduces even more computational cost since each possible split in each dimension is further optimized. The SUHICLUST method was introduced to improve the LOLIMOT space partitioning with the advantages of product space clustering. Due to its use of axis-oblique partitioning, it can generate fuzzy models of the same accuracy as LOLIMOT but with fewer local models. It also builds the models incrementally by splitting the worst performing local region in half. The splitting is done in the direction of the largest eigenvector, which is used to set the initial positions for clusters that are then refined with fuzzy Gustafson-Kessel clustering (local and global). This introduces much computational effort.

Similar to LOLIMOT and HILOMOT, the proposed approach splits the problem space by checking different split options. However, instead of checking each dimension for the split, the split is performed with a hyperplane. The hyperplanes are determined by eigenvectors of the data matrix in the region that has to be divided. However, instead of checking all possible splitting hyperplanes, only the first few are checked according to principle component analysis (PCA) principle. With this, an efficient split procedure is obtained.

In contrast to many established methods, the proposed Hierarchical Fuzzy Model Learning method based on Principal Component Analysis (HiPCA) performs an axis-oblique space partitioning. The computational complexity, given in $\mathcal{O}$-term notation to indicate the number of flops, is relatively low, $\mathcal{O}(c \cdot m^2)$, where c stands for the number of clusters and m for the dimensionality of the feature space. Low complexity is the result of the hyperplane clustering used, which leads to reductions in computational time and memory demand. The algorithm also delivers reproducible modeling results, which is an advantage compared with most other modeling algorithms that produce different results in each run of the algorithm [26]. The complexity of the model is not fixed in advance as it usually is with unsupervised learning methods. Similar as LOLIMOT and SUHICLUST, the proposed HiPCA method has good usability, which is mainly due to its having few tuning parameters. It is possible to generate feasible models by setting only the error threshold. The user can also improve results by selecting different fuzziness factors and different local model approximation algorithms.

The novelty, the highlights of the proposed method, and the differences regarding the SUHICLUST method can be summarized with the following points:

- space splitting based on hyperplanes defined by eigenvectors,
- not all hyperplanes are considered for splitting, only the first few according to the principle component analysis principle,
- calculation of membership degrees is based on the distance to the hyperplanes instead of the distance to the cluster centers,
- no additional clustering is needed.

This paper is organized as follows. In Section I, the introduction to the problem and state-of-the-art is given. After that, in Section II, the methodology of the proposed approach are given. First, the fuzzy model in Takagi-Sugeno form based on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

hyperplane clustering is presented, followed by the splitting of hyperplanes, the hierarchical model tree construction, the detection of optimal splitting hyperplane, and the detection of the next splitting leaf are discussed. In Section III, the benchmark examples and comparisons with other established methods are given, and Section IV concludes the paper.

II. Methodology

A. Takagi-Sugeno fuzzy model based on hyperplane clustering

The Takagi-Sugeno fuzzy model is a universal approximator and capable of describing an arbitrary non-linearity of the observed system by a set of overlapping linear models. Each linear model is defined in its validity region and, together, they are designed to cover the whole data space. The validity regions are defined with the normalized membership functions $\Phi_i,\ i = 1, ..., c$, and the output of the fuzzy model is obtained as the weighed sum of the local linear model outputs $\hat{y}_i$, for which the validity functions are used as weights. Suppose there are c local models (fuzzy rules), the output of the fuzzy model is then given by the following equation:

$$y_m = \sum_{i=1}^{c} \Phi_i(\boldsymbol{z}) y_{m,i}(\boldsymbol{u}) \tag{1}$$

where $\boldsymbol{u}$ stands for the regressor vector $\boldsymbol{u} = [1,\ u_1, ..., u_{n_u}]^T$ which spans the consequent part of the fuzzy rule, which is usually given in a linear affine form as:

$$y_{m,i}(\boldsymbol{u}) = \theta_{i,0} + \theta_{i,1} u_1 + ... + \theta_{i,n_u} u_{n_u}, \tag{2}$$

where $\theta_{i,j},\ \ i = 1, ..., c,\ \ j = 0, ..., n_u$ stands for the parameters of local linear models, which are calculated by weighted least squares (WLS). The regressor vector $\boldsymbol{u}$ consists of all variables that influence the identified process output.

The vector $\boldsymbol{z} = [z_1, ..., z_{n_z}]^T$ spans the premise input space, which, together with other nonlinear parameters, defines the weights, $\Phi_i(\boldsymbol{z}),\ i = 1, ..., c$ of the local linear models defined in Eq. 1. It consists of all variables that best describe the non-linearity of the input-output space.

Both parameters, the linear parameters of the local linear models and the nonlinear parameters of the validity functions, can be defined independently. If the validity functions are defined first, the parameters of the local linear models' θ_{ij} are estimated by using the least squares method. In contrast, the nonlinear parameters could be defined by nonlinear optimization or, what is commonly used, by partitioning of the input-output data space by clustering methods.

The partitioning of the input-output data space is the most important and challenging part in fuzzy model identification. The results of partitioning are the membership functions, which are then normalized to obtain the weights in the form of validity functions as follows:

$$\Phi_i(\boldsymbol{z}) = \frac{\mu_i(\boldsymbol{z})}{\sum_{j=1}^{c} \mu_j(\boldsymbol{z})},\ i = 1, ..., c \tag{3}$$

The membership functions can be of different forms; often triangular, trapezoidal, Gaussian, or Bell forms are used. In our case, the membership functions of the sigmoidal form are used because of simplicity. This means that the membership is a sigmoidal function of the distance between the sample and the splitting hyperplane. The jth splitting hyperplane is defined as follows

$$\boldsymbol{p}_{i,j}^T (\boldsymbol{z} - \boldsymbol{v}_i) = 0, \tag{4}$$

where $\boldsymbol{p}_{i,j}$ stands for jth eigenvector of the covariance matrix $\boldsymbol{\Sigma}_i$, which is orthogonal to the splitting hyperplane, and $\boldsymbol{v}_i$ is the center of the data matrix $\boldsymbol{Z}_i$ of dimensions $N_i \times n_z$, which lies on the splitting hyperplane, and where $\boldsymbol{Z}_i = [\boldsymbol{z}(k)],\ k = 1, ..., N_i$. The matrix of eigenvectors is obtained via the singular value decomposition (SVD) of the covariance matrix $\boldsymbol{\Sigma}_i$, $\boldsymbol{\Sigma}_i = \mathrm{cov}(\boldsymbol{Z}_i)$. The singular value decomposition is given as

$$\boldsymbol{\Sigma}_i = \boldsymbol{P}_i \boldsymbol{\Lambda}_i \boldsymbol{P}_i^T \tag{5}$$

where $\boldsymbol{P}_i$ stands for the matrix of eigenvectors $\boldsymbol{p}_{i,j},\ j = 1, ..., n_z$ with dimension $n_z \times n_z$, and $\boldsymbol{\Lambda}_i$ represents the diagonal matrix of eigenvalues $\boldsymbol{\Lambda}_i = \mathrm{diag}(\lambda_j)$, $j = 1, ..., n_z$ of the same dimension. The mean vector $\boldsymbol{v}_i$ consists of variables' means (the means of the columns in the data matrix $\boldsymbol{Z}_i$). Each eigenvector from matrix $\boldsymbol{P}_i$ together with the mean vector $\boldsymbol{v}_i$ defines one hyperplane, all together n_z hyperplanes in the whole data space of matrix $\boldsymbol{Z}_i$. The splitting hyperplanes are defined by the normal vector on the plane, given by the eigenvectors $\boldsymbol{p}_{i,j},\ j = 1, ..., n_z$ and the mean vector $\boldsymbol{v}_i$.

The membership function can be finally written as a the sigmoidal function of the distance between the sample and the splitting hyperplane, normalized by the corresponding eigenvalue as follows:

$$\mu_i(\boldsymbol{z}) = \frac{1}{1 + \exp\left(\frac{\boldsymbol{p}_{i,j}^T (\boldsymbol{z} - \boldsymbol{v}_i)}{\eta \lambda_j^{\frac{1}{2}}} \right)}, \tag{6}$$

where λ_j defines the width of the membership function and comes from the corresponding eigenvector $\boldsymbol{p}_{i,j}$. The membership function also depends on the user-defined parameter η, which is called the fuzziness parameter. With this parameter, the overlapping between the membership functions is defined. The overlapping of the membership functions affects the smoothness of the approximation; higher overlapping results in a smoother approximation. However, too large overlapping usually leads to a larger identification error.

Note that the term $\boldsymbol{p}_{i,j}^T (\boldsymbol{z} - \boldsymbol{v}_i)$ defines the distance of the data sample $\boldsymbol{z}$ from the hyperplane, as presented in Fig. 1. If the value of $\boldsymbol{p}_{i,j}^T (\boldsymbol{z} - \boldsymbol{v}_i)$ is lower than zero, the sample $\boldsymbol{z}$ belongs to left (otherwise, to the right side of the splitting hyperplane). This hyperplane becomes the line in the case of a 2-dimensional input-output data space, as shown in Fig. 1. In Fig. 1, the unity eigenvectors $\boldsymbol{p}_{i,j},\ j = 1, 2$ are scaled to show the variance of the data in each direction of eigenvectors, i.e., $\boldsymbol{g}_j,\ j = 1, 2$ stands for the principal component of the data and is defined as $\lambda_j^{\frac{1}{2}} \boldsymbol{p}_{i,j},\ j = 1, 2$.

B. Splitting hyperplanes and hierarchical model tree

The presented splitting results in a hierarchical model tree. The basic idea of the proposed method is to split the data inside the input-output data space by hyperplanes into smaller

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

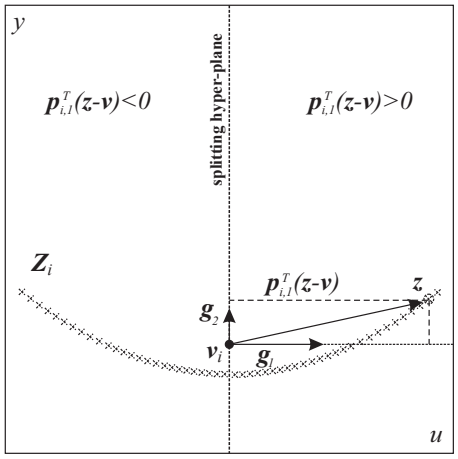

Fig. 1. Data-space partitioning by hyperplane splitting.

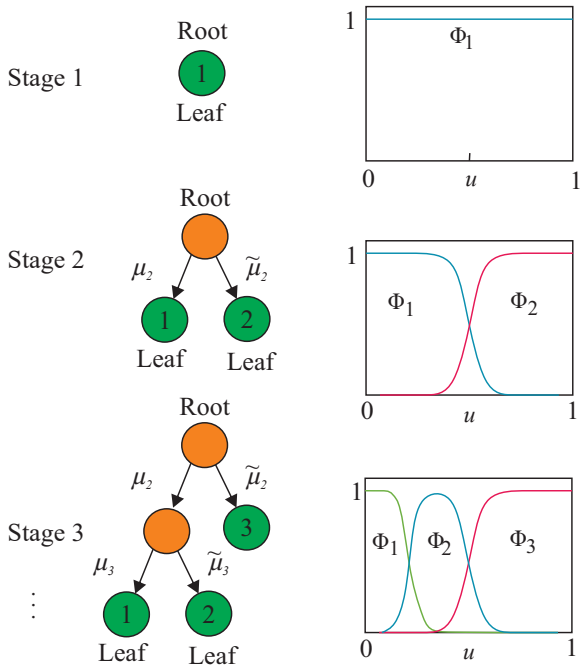

Fig. 2. Hierarchical model principle. The validity functions of each leaf results from the multiplication of all sigmoid functions along the path.

subspaces, which are then modelled by local linear models. Splitting the data space into smaller subspaces and attempting to find the local models in the form of linear models means that we are attempting to find the set of local models that will together minimize the unexplained variance of the measured data samples. In the following, the mechanism of splitting is presented in more detail. The model structure is presented in Fig. 2.

The algorithm starts with one leaf containing all the data (covering the entire problem space). Its validity function (normalized membership degree) equals one throughout the entire region. Next, in the splitting stage 2, this leaf is split in half (leaves 1 and 2) regarding the hyperplane, which is defined by the eigenvectors and the center of the covariance matrix for the whole data set. If the value $\boldsymbol{p}_{i,j}^T(\boldsymbol{z}-\boldsymbol{v}_i)$, for the sample $\boldsymbol{z}$ is lower than zero the sample belongs to one leaf (otherwise, to the other leaf). The left leaf is presented by membership functions μ_i and the right one by the complementary function $\tilde{\mu}_i$. Fig. 2 represents three stages of the hierarchical splitting, which results in three validity functions, because in each stage one leaf is divided into two new leaves. In total, one leaf is added to the model.

The data is then, according to the defined hyperplane, divided between the two new leaves. Using these data, the local model for each leaf is identified using the least squares method. At each splitting stage, a green leaf with the poorest performance is selected and split. In the presented case, in Fig. 1, these are the orange root in splitting stage 2 and the left orange leaf from stage 3. The splitting of the leaves is performed until a predefined tolerance of the whole model approximation is archived. The membership degrees of clusters $\mu_i^B,\ i=1,...,c$ are calculated as the product of all sigmoid functions μ_j or $\tilde{\mu}_j$ along the traveling path (the branch) from the ending leaves (i-th green leaves) to the root leaf as follows:

$$\mu_i^B(\boldsymbol{z}) = \prod_{j=1}^{B_i} \begin{cases} \mu_j(\boldsymbol{z}), & \boldsymbol{p}_j^T(\boldsymbol{z}(k)-\boldsymbol{v}_j) \le 0. \\ \tilde{\mu}_j(\boldsymbol{z}), & \text{otherwise.} \end{cases} \tag{7}$$

where B_i stands for the number of leaves from i-th green leaf to the root leaf. The validity functions $\Phi_i(\boldsymbol{z}),\ i=1,...,c$ are defined as the normalized membership functions

$$\Phi_i(\boldsymbol{z}) = \frac{\mu_i^B(\boldsymbol{z})}{\sum_{j=1}^{c}\mu_j^B(\boldsymbol{z})},\ i=1,...,c \tag{8}$$

For example, in Fig. 2, there are three final green leaves at the splitting stage 3, which means three rules with three validity functions $\Phi_1(\boldsymbol{z}), \Phi_2(\boldsymbol{z})$ and $\Phi_3(\boldsymbol{z})$. The membership degree $\mu_1^B(\boldsymbol{z})$ is defined as the product of all membership degrees from the green leaf number 1 to the root leaf. The number of leaves along this traveling path is $B_1=2$. The membership functions are as follows: $\mu_1^B(\boldsymbol{z}) = \mu_3(\boldsymbol{z})\mu_2(\boldsymbol{z})$, $\mu_2^B(\boldsymbol{z}) = \tilde{\mu}_3(\boldsymbol{z})\mu_2(\boldsymbol{z})$, and $\mu_3^B(\boldsymbol{z}) = \tilde{\mu}_2(\boldsymbol{z})$. The normalized membership degree or the validity function $\Phi_1(\boldsymbol{z})$ is then defined as follows:

$$\Phi_1(\boldsymbol{z}) = \frac{\mu_3(\boldsymbol{z})\mu_2(\boldsymbol{z})}{\mu_3(\boldsymbol{z})\mu_2(\boldsymbol{z}) + \tilde{\mu}_3(\boldsymbol{z})\mu_2(\boldsymbol{z}) + \tilde{\mu}_2(\boldsymbol{z})}. \tag{9}$$

The hierarchical model tree from Fig. 2 is, in the form of a fuzzy model, described as follows

$$R_i:\ \textbf{if}\ \boldsymbol{z}(k)\ \textbf{is}\ \Phi_i(\boldsymbol{z}(k))\ \textbf{then}\ y_m(k) = y_{m,i}(k),\ i=1,2,3 \tag{10}$$

Using the sigmoid membership functions is only one way of handling the transitions. There are also other possibilities (e.g., linear or Gaussian functions over the area of transition).

C. Detection of optimal splitting hyperplane

In Fig. 1, it is shown that the largest principal vector $\boldsymbol{g}_1$, the vector with the largest variance, points approximately in the direction of the model, at least much better than the vector with lower variance $\boldsymbol{g}_2$. The idea of splitting is to find as many subspaces as needed to describe the most important variance of the data, and only a small latent variance will remain. This latent variance, which remains undescribed, is called the noise.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

Based on the assumption that the largest eigenvector points approximately in the direction of model, the idea of the HiPCA is to split the data in the direction of the vectors with the largest variance. This means that the eigenvector with the largest eigenvalue (i.e., the eigenvector $\boldsymbol{p}_1$ in Fig. 1) is the normal vector to the splitting hyperplane. However, a 2-D example can quickly be developed in which the data is distributed so that both eigenvalues are the same. Therefore, the splitting procedure cannot be done only based on the largest eigenvector. The HiPCA partitioning procedure considers splitting the hyperplanes perpendicular to the directions of the highest variances of the data, which means in the direction of the largest eigenvectors that are considered as candidates for the normal to the splitting hyperplanes.

To find all possible candidates for splitting, the data set of raw measurement $\boldsymbol{Z}_i$ of dimension $N_i \times n_u + 1$, defined as $\tilde{\boldsymbol{z}}(k) = [\tilde{\boldsymbol{u}}(k) \;\; \tilde{y}(k)]$, $k = 1, ..., N_i$, where $\tilde{\boldsymbol{u}}(k) = [\tilde{u}_1(k), ..., \tilde{u}_{n_u}(k)]^T$, should be first centered by mean of each variable in the data matrix $\boldsymbol{Z}_i$, where m_i is the mean of variable $\tilde{u}_i$ and $i = 1, ..., n_u$ and m_y is the mean of y, and scaled ($K_i = \max|\tilde{u}_i(k) - m_i|, \quad i = 1, ..., n_u$ and $K_y = \max|\tilde{y}(k) - m_y|$), i.e., the variables are centered by mean values of the raw variables and scaled by absolute maximal deviation of variables from the mean value. In that sense, all the values of normalized variables are in the interval between -1 and 1. The normalized data matrix is then described as follows:

$$\boldsymbol{Z}_i = \begin{bmatrix} u_1(1) & \dots & u_{n_u}(1) & y(1) \\ u_1(2) & \dots & u_{n_u}(2) & y(2) \\ \vdots & \ddots & \vdots & \vdots \\ u_1(N) & \dots & u_{n_u}(N) & y(N) \end{bmatrix} \tag{11}$$

and the covariance matrix, which describes the distribution of the data, as follows:

$$\boldsymbol{\Sigma}_i = \frac{1}{N_i - 1} \boldsymbol{Z}_i^T \boldsymbol{Z}_i \tag{12}$$

Next, the singular value decomposition of the covariance matrix is calculated, and the matrices of eigenvectors and eigenvalues matrices are obtained as proposed in Eq. 5.

Once the eigenvectors and eigenvalues are obtained, the number of possible splitting hyperplanes r is calculated as the minimal number of eigenvalues to satisfy the condition in Eq. 13:

$$r = \arg\min_n \left(\frac{\sum_{k=1}^{n} \lambda_k}{\sum_{j=1}^{n_z} \lambda_j} \geq \delta, \; n \in [1, n_z] \right) \tag{13}$$

where the threshold δ is usually set to 0.95, meaning that only the eigenvectors that explain the 95% of whole data variance are considered as a splitting candidates. This means that in each splitting stage s, the worst leaf from stage $s-1$ is split into two new leaves (right leaf R and left leaf L), and the splitting can be done in r different directions. Accordingly, the data set from the previous leaf is divided between new leaf pairs as follows:

$$\boldsymbol{Z}_{s,j}^L = \left\{ \boldsymbol{z}(k) \in \boldsymbol{Z}_s : \boldsymbol{p}_{s,j}^T (\boldsymbol{z}(k) - \boldsymbol{v}_s) \leq 0; k = 1, ...N_s \right\} \tag{14}$$

$$\boldsymbol{Z}_{s,j}^R = \left\{ \boldsymbol{z}(k) \in \boldsymbol{Z}_s : \boldsymbol{p}_{s,j}^T (\boldsymbol{z}(k) - \boldsymbol{v}_s) > 0; k = 1, ...N_s \right\} \tag{15}$$

where $\boldsymbol{Z}_s$ is data set of the previous leaf that is being split, N_s is the number of samples in that leaf, $\boldsymbol{p}_{s,j}, \; j = 1, .., r$ is the chosen eigenvector and $\boldsymbol{v}_s$ is the center that defines the jth splitting hyperplane; the upper index L defines the left and index R the right leaf. For each new leaf, a new local model is identified using the least squares method (LSM).

Having r possible splitting, we also obtain r possible different fuzzy model outputs $y_m^j, \; j = 1, ..., r$ in the normalized data space. Among these r different models, the model that approximates the measured data the best is chosen, i.e., the model that has the smallest normalized root mean square error (NRMSE) between the measured output data and the simulated model output given in Eq. 16,

$$\Gamma_j = \left(\frac{\sum_{k=1}^{N} (y(k) - y_m^j(k))^2}{\sum_{k=1}^{N} (y(k) - \overline{y})^2} \right)^{\frac{1}{2}}, j = 1, ..., r \qquad \overline{y} = \frac{1}{N} \sum_{k=1}^{N} y(k) \tag{16}$$

where $y_m^j(k)$ is the jth model output and $y(k)$ is the normalized output at sample k. This means that the real model output in measured data space is re-scaled and shifted as follows: $\tilde{y}_m^j(k) = K_y \cdot y_m^j(k) + m_y, \; j = 1, ..., r$. In general, different types of error can be used. After calculation of NRMSE for all possible splitting, the split and the model with the smallest NRMSE and the corresponding model is selected as follows:

$$h = \arg\min_j \Gamma_j \tag{17}$$

This gives the splitting hyperplane parameters ($\boldsymbol{p}_{s,h}$, $\boldsymbol{v}_s$ and $\lambda_{s,h}$) and new leaves, i.e., two new leaves in stage 3, and the leaves numbers 1 and 2 as shown in Fig.2.

Notice that usually the model of the process is used to predict the output; this means that only the input variables are known. However, to calculate the validity functions of the fuzzy model, the whole input-output data vector $\boldsymbol{z}(k)$ should be known. This is done with the principle of projection in which the last element of $\boldsymbol{z}(k)$ is substituted with the last component of the center vector $\boldsymbol{v}_s$.

D. Detection of next splitting leaf

When all validity functions are defined, the error of the whole current fuzzy model Γ is calculated as follows:

$$\Gamma = \sum_{i=1}^{c} \sum_{k=1}^{N} \Phi_i(\boldsymbol{z}(k)) (y(k) - y_{m,i}^h(k))^2 \; . \tag{18}$$

The whole model error in Γ is then divided among the end-leaves. The simplest and most effective one is the local sum of squared errors, i.e., standard deviation of local model [25], calculated as:

$$e_i = \sum_{k=1}^{N} \Phi_i(\boldsymbol{z}(k)) (y(k) - y_{m,i}^h(k))^2, i = 1, .., c \tag{19}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

where c is the number of local models and y_m^h is the new model output.

The leaf with the highest error e_i is chosen for the further split as follows:

$$w = \arg\max_i e_i \ . \tag{20}$$

In a case of Fig. 2, in stage 2, this leaf is leaf number 1. The procedure is repeated until a suitable approximation is reached (or until the maximal number of regions is reached). The algorithm also stops if there are no leaves left to split. The leaves are not allowed to be split if their maximal eigenvalue is below a certain threshold, which is related to the variance of noise (four times of the noise variance) or if there are not enough data points belonging to that leaf.

The pseudo-code for the HiPCA algorithm in the normalized data domain is given in Alg. 1.

Algorithm 1 HiPCA learning algorithm

Define tuning parameters: η, c_{max}, Γ_{max}, δ
Initialize: $c = 1$
Estimation of $\theta_{i,j}$ for linear model ($c = 1$)
loop:
  Computation of fuzzy model output from Eq. 1
    Computation of membership degrees from Eqs. 6, 7
    Normalization of membership from Eq. 8
    Calculate model error Γ from Eq. 18
  if $\Gamma \leq \Gamma_{max}$ **or** $c \geq c_{max}$ **return** model and **stop**
  Computation of e_i from Eq. 19, $i = 1, ..., c$
  Select the cluster w with the highest error from Eq. 20
  Calculate $\boldsymbol{v}_w$ and $\boldsymbol{\Sigma}_w$
  Calculate $\boldsymbol{P}_w$ and $\boldsymbol{\Lambda}_w$ from Eq. 5
  Find smallest r that satisfies Eq. 13
  $c \leftarrow c + 1$
  for $j = 1 : r$ (test possible splits)
    Split the data of w-th cluster to $\boldsymbol{Z}_{w,j}^L$ and $\boldsymbol{Z}_{w,j}^R$ as given in Eqs. 14, 15
    Estimation of local model parameters for clusters $\boldsymbol{Z}_{w,j}^L$ and $\boldsymbol{Z}_{w,j}^L$ by WLS
    Computation of fuzzy model output from Eq. 1
    Calculate model error Γ_j from Eq. 18
  end
  Select the best splitting hyperplane h as given in Eq. 17
  Split the wth cluster into two new clusters using hyperplane h
goto loop

The variable $\boldsymbol{Z}_w$ used in Alg. 1 represents the data matrix of the wth leaf or subspace; w represents the index of the splitting leaf; c represents the number of green end-leaves; $\boldsymbol{P}_w$ and $\boldsymbol{\Lambda}_w$ represent the eigenvector matrix and eigenvalue matrix of the splitting leaf, respectively; Γ^m represents the current model error; e_j represents the error of the jth end-leaf; $\boldsymbol{Z}_{w,j}^L$ and $\boldsymbol{Z}_{w,j}^R$ represent the data matrix of the new left and right leaf obtained by splitting the previous worst leaf with the jth hyperplane; $\boldsymbol{p}_{w,j}$ and $\boldsymbol{v}_w$ defines the jth splitting hyperplane; h defines the index of best split.

III. Benchmark examples and comparisons

The proposed method is compared to other methods like LOLIMOT (LOLI) [31], [33], HILOMOT [33] (HILO), SUHI-CLUST [26], [24], [25] (SUHI), *fuzzyid* [34], *zofuzid* [34] and *hinging-hyperplanes* (HP) [10]. The method was also compared to the methods implemented in the *sk-learn* [35] package: the popular Random Forest Regressor method (*RF*), the Support Vector Machine (*SVR*), the Nearest Neighborhood Regression Tree *KNNT*, the AdaBoost Tree (*ADAB*), and the Extra Tree Regressor (*ETR*). The methods/toolboxes used are implemented in different programming environments. The *sk-learn* is implemented in Python; HILOMOT, *fuzzyid* [34], *zofuzid* [34] and hinging-hyperplanes (HP) in Matlab; SUHI-CLUST and HiPCA in $C\#$. However, the Matlab implementation of LOLIMOT does not allow changing the fuzziness factor and does not consider the possibility of estimating all local models' parameters in the new iteration. This can sometimes lead to better results; therefore, the LOLIMOT was also re-implemented in the $C\#$ environment, giving the method more flexibility (LOLI $C\#$). The software package for the HiPCA method can be downloaded from [36]. The performances of methods are compared on the static and dynamic examples presented in the following subsections. The methods were tested on Lenovo T570 laptop with an i7, 2.9 GHz processor and 16GB RAM. The execution time is measured from the start of method execution to the final result. The provided execution times are measured from the user's point of view and only serve as an orientation about the computational complexity of the algorithms.

All the results are presented in the form of tables, in which the first column of the table represents the method name. In the second column, the number of generated local models follows (in the case of SVR, the number of support vectors). The third column represent the error on the learning set and the fourth the error on the testing set. The fifth column represents the number of assembled models (#EM). This number is equal one for most methods that are not based on the ensembles principle. Ensemble-based methods have more differently learned nonlinear models from which the final output estimation is made either by a voting principle or simply by averaging the outputs of all models. The number of local models for the assembled methods is calculated as the average number of local models of all generated end-models. The last column represents the approximate a needed to calculate the model. In all tables, the dividing line is drawn to make the distinction between fuzzy and non-fuzzy approaches. These non-fuzzy approaches are implemented in *sk-learn* package.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

Method	#LMs	Γ_l	Γ_t	#EM	t[s]
HiPCA	**5**	**0.0278**	**0.0282**	**1**	$< \mathbf{1}$
LOLI	20	0.0495	0.0686	1	≈ 3
LOLI $C\#$	11	0.0452	0.0528	1	≈ 1
HILO	6	0.0489	0.0502	1	≈ 4
SUHI	9	0.0474	0.0524	1	≈ 4
fuzzyid	25	0.0567	NaN/0.0534	1	< 1
zofuzid	16	0.0818	NaN/0.0780	1	< 1
HP	8	0.0645	0.0648	1	≈ 5
KNNT	**31**	**0**	0.054	1	< 1
ADAB	357.65	0.012	0.086	20	< 1
ETR	866.5	**0**	0.049	10	< 1
RF	32	0.183	0.232	10	< 1
SVR	175	0.006	**0.007**	1	≈ 119

TABLE I
RESULTS FOR MARS PROBLEM

Method	#LMs	Γ_l	Γ_t	#EM	t[s]
HiPCA	**25**	**0.0461**	0.0635	1	≈ 14
LOLI	31	0.1362	0.1562	1	≈ 6
LOLI $C\#$	31	0.0498	0.0649	1	≈ 20
HILO	30	0.0478	0.0618	1	≈ 26
SUHI	29	0.0477	**0.0577**	1	≈ 197
fuzzyid	81	0.059	0.0771	1	≈ 17
zofuzid	81	0.2451	0.2523	1	< 1
HP	76	0.1404	0.1796	1	≈ 58
KNNT	**31**	**0**	0.099	1	< 1
ADAB	521.2	0.051	0.116	10	< 1
ETR	863.8	0.073	0.094	10	< 1
RF	235.5	0.092	0.149	10	< 15
SVR	925	0.005	**0.006**	1	≈ 2926

TABLE II
RESULTS FOR MG SERIES PREDICTION PROBLEM

A. Static examples

Three examples were chosen for comparison of static model problem: the MARS 1 problem [37], the prediction of MackeyGlass time series [38] and Hyperbola-benchmark [25].

1) The MARS 1 examples: In this example, the goal is to model the function:

$$y = \frac{2e^{f_1}}{e^{f_2} + e^{f_3}} \tag{21}$$

where $f_1 = 8((u_1-0.5)^2+(u_2-0.5)^2)$, $f_2 = 8((u_1-0.2)^2+(u_2-0.7)^2)$ and $f_3 = 8((u_1-0.7)^2+(u_2-0.2)^2)$.

The size of the learning set was 900 samples, for which u_1 and u_2 were randomly generated from a uniform distribution. The validation data was also 900 samples; however, in this case, u_1 and u_2 were equally distributed from zero to one, creating a grid. The goal was to achieve an NRMSE of 0.05. The global least squares method was used to estimate the local model parameters for all fuzzy methods. Detailed settings of the methods and scripts can be found in [36]. The obtained results are presented in Table I.

As seen from the results table, the methods implemented in the *sk-learn* package are very fast; however, they usually need many local models to achieve good accuracy. For example, the ETR method created on average 866 local models (or end leaves) for the data set of 900 samples. Also, the difference between the learning and testing NRMSE is much higher than obtained by other methods that are usually used in control engineering. Note that the SVR learning time is much higher than by other methods. The reason for that is that the *grid search* method was used, which checks multiple possible settings. In this test, the *fuzzyid* and *zofuzid* produced the *NaN* NRMSE since few samples were estimated as *NaN*-s. Along with the *NaN* value, the NRMSE value is also given, which is obtained not considering *NaN*-s. The HiPCA method obtained the model in less than a second and with only 5 local models. The tuning parameters of HiPCA are: the termination criteria defined by the maximal NRMSE equal to $\Gamma_{max} = 0.05$, the maximal number of local models defined as $c_{max} = 20$, and the fuzziness parameter $\eta = 1.5$.

2) MackeyGlass time series: The chaotic time series [38] is generated from the MackeyGlass (MG) differential delay equation defined by the following equation:

$$z(t) = \frac{0.2z(t-\tau)}{1+z^{10}(t-\tau)} - 0.1z(t) \tag{22}$$

where the initial condition and τ are set as $z(0) = 1.2$ and $\tau = 17$. The aim is to use past values of z to predict a future value of z. The value of the signal is predicted 85 steps ahead, based on the values of the signal at the current moment, and 6, 12, and 18 steps back:

$$\begin{aligned} y(k) &= [z(k+85)] \\ \boldsymbol{u}(k) &= [z(k-18)\ z(k-12)\ z(k-6)\ z(k)] \end{aligned} \tag{23}$$

The training set is comprised of data points in the interval $k \in [201, 3200]$ and the validation set from points in the interval $k \in [5001, 5500]$. The obtained results are presented in Table II. In this case, the *fuzzyid* and *zofuzid* produced valid test results. Again, the methods from *sk-learn* were very fast but did not perform as well as HiPCA or SUHICLUST on the test results. Again, judging from the NRMSE, the best performance was by SVR obtained via *grid search* but the number of local models is almost one-third of the whole learning data set size. The best NRMSE was, in this case, obtained with the SUHICLUST method; however, that method produced four local models more than HiPCA did, and the learning time with SUHICLUST was considerably longer. In this example, the tuning parameters of HiPCA are: the termination criteria defined by the maximal NRMSE equal to $\Gamma_{max} = 0.05$, the maximal number of local models defined as $c_{max} = 20$ and the fuzziness parameter $\eta = 0.6$.

3) Hyperbola example: In this example, the goal is to model the Hyperbola function:

$$y = \frac{1}{0.1 + \frac{1}{p}\sum_{i=1}^{p}(1-u_i)} \tag{24}$$

The tests were made for 1-, 4-, 7-, and 10-D input space. The inputs were randomly generated from an interval $[0, 1]$. For learning, 900 samples were used and for testing 2000 samples were used. The number of generated local models is presented in Table III. For the one-dimensional problem, all methods perform reasonably well. Increasing the dimension, the results of the methods *sk-learn* library decrease. The NRMSE on the learning set increases substantially together with the NRMSE on testing data. The *fuzzyid* and *zofuzid* had a substantial

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

	1-D	4-D	7-D	10-D
Method	#LMs / #EM			
HiPCA	**4/1**	**3/1**	**3/1**	**3/1**
LOLI	5/1	22/1	32/1	29/1
LOLI $C\#$	4/1	10/1	14/1	18/1
HILO	4/1	4/1	4/1	4/1
SUHI	5/1	5/1	5/1	3/1
fuzzyid	5/1	16/1	128/1	/
zofuzid	5/1	16/1	128/1	1024/1
HP	16/1	4/1	2/1	3/1
KNNT	63/1	63/1	63/1	63/1
ADAB	112.8/10	112/10	107.5/10	104/10
ETR	765.5/8	895.2/8	897/8	897.2/8
RF	90.3/10	89.9/10	90/10	91/10
SVR	112.8/10	112/10	107.5/10	104/10

TABLE III
RESULTS FOR HYPERBOLA PROBLEM (NUMBER OF LOCAL MODELS)

Method	#LMs	Γ_l	Γ_t	#EM	t[s]
HiPCA	**4**	0.1194	**0.1578**	**1**	$< \mathbf{1}$
LOLI	12	0.1157	0.3595	1	≈ 3
LOLI $C\#$	7	0.1194	0.3375	1	≈ 1
HILO	7	0.5750	0.4250	1	≈ 11
SUHI	6	**0.1079**	0.1666	1	< 1
KNNT	15	0.182	0.464	1	< 1
ADAB	189.9	0.222	0.482	10	< 1
ETR	357	**0.004**	0.326	10	< 1
RF	**8.4**	0.633	0.622	5	≈ 1
SVR	355	0.123	**0.18**	1	≈ 31

TABLE IV
RESULTS FOR CED

problem with the dimensionality. The *fuzzyid* method failed to generate a model for a 10-D problem, while the error of the model generated with *zofuzid* had a learning error around 0 and a testing error of 1.6. Also, the control of number of local model was impossible for these two methods. Since they use grid partitioning, the number of local models increase exponentially with the dimension of the data space.

An interesting observation can be made by examining the number of generated local models. It can be seen that the number of local models increases with the space dimension with the LOLIMOT method while with the HiPCA, SUHICLUST, and HP methods the number of local models decreases or at least stays the same. The reason for decreasing the number of local models is in the variance of the output that decreases with the dimension. The output variance for the 1-D problem is 3.4, for 4-D 0.28, for 7-D 0.11 and for a 10-D problem 0.08. The decrease of variance means that the span of output is lower and, therefore, the nonlinearity also decreases. This means that fewer local models are needed to approximate the model. In all hyperbola examples, the tuning parameters of HiPCA are: the termination criteria defined by the maximal NRMSE equal to $\Gamma_{max} = 0.05$, the maximal number of local models defined as $c_{max} = 20$ and the fuzziness parameter $\eta = 1.0$.

B. Dynamic examples

Three examples were chosen for dynamic modeling problem: Silver Box example (SB) [39], the Cascade Tanks Example (TCT) [39], and the Coupled Electrical drive example (CED) [39]. As seen from previous static examples, the *fuzzyid* and *zofuzid* have problems with high dimensional examples; therefore, they were omitted in the dynamical model testing since the regressors have higher dimensions.

1) Coupled electric drives: The coupled electric drives system consists of two electric motors that are used to drive a pulley by using a flexible belt. The pulley is held by a spring, resulting in a lightly damped dynamic mode. The electric drives can be individually controlled, allowing the tension and the speed of the belt to be simultaneously controlled. The problem in this case is to identify the relation between the process input, which is the sum of the voltages applied to the motors, and the output, which is the pulley velocity [39]. According to [40], the regressor in the form of $\boldsymbol{u}(k) = [1, y(k-1), y(k-4), y(k-6), y(k-7), y(k-8), y(k-9), u(k-1), u(k-4), u(k-6)]$ was chosen. The space for partitioning was comprised of inputs and outputs delayed for 1, 3, 6, and 9 samples. The training data set has 374 samples while the testing data set has 126 samples. The results of identification are presented in Table IV. In this example, the target NRMSE error for the HILO, SUHI, LOLI, and HiPCA method was set to 0.12. The HP method failed to produce any model. The HiPCA method reached the threshold with only four local models and has the lowest test NRMSE. While the HILOMOT produced good models for static examples, it failed to produce a good model for the presented dynamic example. The learning stopped because the algorithm could no longer decrease the loss function. The tuning parameters of HiPCA are: the termination criteria defined by the maximal NRMSE equal to $\Gamma_{max} = 0.12$, the maximal number of local models defined as $c_{max} = 20$, and the fuzziness parameter $\eta = 4$.

2) Two cascaded tanks: This process is a liquid level control system consisting of two cascaded tanks with free outlets fed by a pump. The liquid (the demineralized water) is transported by the pump to the upper of the two tanks. The input signal to the process is the voltage applied to the pump $u(t)$, and the two output signals consist of measurements of the water levels of the tanks h_1 and h_2. Since the outlets are open, and since the tanks are relatively high, the result is a significantly non-linear dynamic that varies with the level of water [39]. The goal here is to identify the level h_2. According to [40], the regressor in the form of $\boldsymbol{u}(k) = [1, h_2(k-4), h_2(k-1), h_1(k-4), u(k-2), u(k-4)]$ was chosen. The space for partitioning was defined as $\boldsymbol{z}(k) = [h_2(k), h_2(k-1), h_1(k-1), h_1(k-4), u(k-1), u(k-4)]$. The training data set has 1500 samples while the testing data set has 1000 samples. The results of modeling are presented in Table V. The error threshold for the HILO, SUHI, LOLI, and HiPCA methods was set to 0.05. Again, the HP method failed to produce a model due to singularity issues. In this example, the HILOMOT produced a valid model with the same number of local models as HiPCA and SUHICLUST did, with the HiPCA having a slightly better test NRMSE value. In this example, the tuning parameters of HiPCA are: the termination criteria defined by the maximal NRMSE equal to $\Gamma_{max} = 0.05$, the maximal number of local models defined as $c_{max} = 20$ and the fuzziness parameter $\eta = 1.5$.

3) Silver box: This example concerns an electronic nonlinear feedback laboratory system (denoted as "the silver

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2019.2931243, IEEE Transactions on Industrial Electronics

Method	#LMs	Γ_l	Γ_t	#EM	t[s]
HiPCA	**4**	0.04711	0.0465	1	< 1
LOLI	7	**0.0438**	**0.0406**	1	≈ 3
LOLI $C\#$	5	0.0463	0.0517	1	< 1
HILO	**4**	0.0498	0.0518	1	≈ 8
SUHI	**4**	0.0485	0.0533	1	≈ 8
KNNT	**63**	0.035	0.11	1	< 1
ADAB	794.7	0.06	0.197	10	< 1
ETR	1433.9	**0**	0.194	10	< 1
RF	27.8	1.054	1.162	5	≈ 1
SVR	1248	0.039	**0.052**	1	2459

TABLE V
RESULTS FOR TCT

Method	#LMs	Γ_l	Γ_t	#EM	t[s]
HiPCA	5	0.0415	0.0336	1	≈ 2
LOLI	22	0.048	0.1337	1	≈ 81
LOLI $C\#$	**4**	**0.0289**	**0.03**	1	≈ 1
HILO	22	1.620	1.6583	1	≈ 401
SUHI	5	0.0445	0.0421	1	≈ 7
HP	8	0.1487	0.1466	1	≈ 170
KNNT	511	0.292	0.258	1	< 1
ADAB	4413.4	0.063	0.256	10	≈ 1
ETR	9499.5	0.114	0.226	10	< 1
RF	**32**	0.653	0.662	10	< 1
SVR	5444	**0.037**	**0.03**	1	≈ 386

TABLE VI
RESULTS FOR SB

box"). Analogue electrical circuitry is used to generate data representing a nonlinear mechanical resonating system with a moving mass, a viscous damping, and a nonlinear spring. The purpose of the electrical circuit is to relate the displacement $y(k)$ to the force $u(k)$. The nonlinear spring is described by using a static position-dependent stiffness [39]. According to [40], the regressor in the form of $\boldsymbol{u}(k) = [1, y(k-1), y(k-2), y(k-4), u(k-1), u(k-2), u(k-3)]$ was chosen. The space for partitioning was the same as the regressor, but first element (1) was replaced with process output $y(k)$. The training data set has 10000 samples while the testing data set has 4000 samples. The results of modeling are presented in Table VI. In this example, the best results were obtained with the LOLIMOT$C\#$ method. While the HILOMOT method failed to produce a good model, the SUHICLUST and HiPCA needed one local model more than LOLIMOT$C\#$ did to get the error below the threshold of 0.05. The reason that the results for this example are better with the LOLIMOT method than with HiPCA and SUHICLUST may be in the fact that both HiPCA and SUHICLUST split the space based on the data covariance matrix. If the partition space is incorrectly chosen (e.g., linearly dependent inputs), both methods might have problems obtaining a good model. In this example, the tuning parameters of HiPCA are: the termination criteria defined by the maximal NRMSE equal to $\Gamma_{max} = 0.05$, the maximal number of local models defined as $c_{max} = 20$ and the fuzziness parameter $\eta = 1.5$.

IV. CONCLUSION

In this paper, hierarchical fuzzy space partitioning based on hyperplanes for Takagi-Sugeno fuzzy model identification is proposed. The method was tested and compared to the existing and established fuzzy model and regression tree learning methods. The comparisons were performed on three static and three dynamic examples. The results show that the method is relatively fast and can achieve the same or even better results than established methods can (e.g., LOLIMOT, SUHICLUST). Except in one example (Silver Box), the presented method outperformed the existing fuzzy learning methods in at least one measured category. It usually generates fewer local models to achieve the desired accuracy. Compared to the methods implemented in the *sk-learn* library, the tuning of the presented method is easier and more straightforward. Very significantly, the proposed method, in contrast to many other methods, delivers reproducible results and does not depend on stochastic initialization. The performance of the method is studied and validated using a number of analytical examples, benchmark problems from the literature, and real-process data. The obtained results are highly promising.

The presented method splits the data based on the covariance matrix of the data. Therefore, a bad selection of the partitioning space and the distribution of the data will affect the results, as is usual with all data mining methods. The future development will go into these two directions. The preliminary study that was made showed that moving the splitting hyperplane in the direction of its normal can further improve the results, especially when dealing with unbalanced data. Preliminary results also show that removing the variables that have a low correlation with the target output in a certain fuzzy region and removing inputs that are highly correlated to one and other in a certain fuzzy region improves the results significantly.

ACKNOWLEDGEMENT

This work has been supported by Slovenian Research Agency with the Research Program P2-0219.

REFERENCES

[1] T. A. Johanson and R. Murray-Smith, *Operating regime approach to nonlinear modeling and control*, T. A. Johanson and R. Murray-Smith, Eds. U.K.: Taylor Francis, 1981.

[2] T. Takagi and M. Sugeno, "A fuzzy identification of systems and its applications to modelling and control," *IEEE Trans. on Syst., Man and Cyber.*, vol. 15, no. 1, pp. 116–132, 1985.

[3] R. Babuška and H. B. Verbruggen, "An overview of fuzzy modeling for control," *Control Eng. Practice*, vol. 4, no. 11, pp. 1593–1606, 1996.

[4] R. H. Abiyev and S. Abizade, "Diagnosing parkinson's diseases using fuzzy neural system," *Computational and Mathematical Methods in Medicine*, vol. 2016, no. http://dx.doi.org/10.1155/2016/1267919, p. 9, 2016.

[5] L. Breiman, "Hinging hyperplanes for regression, classification and function approximation," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 311 – 325, 1993.

[6] C. Wen, X. J. S. Wang, and X. Ma, "Identification of dynamic systems using piecewise-affine basis function models," *Technical Report LiTH-ISY-R-1809, Department of Electrical Engineering, Linkoping University, Linkoping, Sweden*, vol. 43, no. 10, pp. 1824 – 1831, 2007.

[7] S. Ernst, "Hinging hyperplane trees for approximation and identification decision and control," *Proceedings of the 37th IEEE Conference*, vol. 2, pp. 1266 – 1271, 1998.

[8] P. Pucar and J. Sjoberg, "Parameterization and conditioning of hinging hyperplane models," *Technical Report LiTH-ISY-R-1809, Department of Electrical Engineering, Linkoping University, Linkoping, Sweden*, 1995.

[9] J. Roll, A. Bemporad, and L. Ljung, "Identification of piecewise affine systems via mixed-integer programming," *Automatica*, vol. 40, pp. 37 – 50, 2004.

[10] T. Kenesei and J. Abonyi, "Hinging hyperplane based regression tree identified by fuzzy clustering and its application," *Applied Soft Computing*, vol. 13, pp. 782 – 792, 2013.

[11] R. Hathaway and J. Bezdek, "Switching regression models and fuzzy clustering," *IEEE Transactions on fuzzy systems*, vol. 1, no. 3, pp. 195 – 204, 1993.

[12] G. Klančar and I. Škrjanc, "Evolving principal component clustering with a low run-time complexity for lrf data mapping," *Applied Soft Computing*, vol. 35, pp. 349 – 358, 2015.

[13] R. Vidal, Y. Ma, S. Soatto, and S. Sastry, "Two-view multibody structure from motion," *International Journal of Computer Vision*, vol. 68, no. 1, pp. 7–25, 2006.

[14] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "An algebraic geometric approach to the identification of a class of linear hybrid systems," *In IEEE Conference on Decision and Control*, pp. 167–172, 2003.

[15] L. Bako, "Identification of switched linear systems via sparse optimization," *Automatica*, vol. 47, no. 4, pp. 668–677, 2011.

[16] W. Zou, C. Li, and N. Zhang, "A ts fuzzy model identification approach based on a modified inter type-2 frcm algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1104–1113, 2018.

[17] M. Ferdaus, M. Pratama, S. Anavatti, and M. Garratt, "Palm: An incremental construction of hyperplanes for data stream regression," *IEEE Transactions on Fuzzy Systems*, vol. In Press, 2019.

[18] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (gpca)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1–15, 2005.

[19] G. Chen and G. Lerman, "Spectral curvature clustering (scc)," *International Journal of Computer Vision*, vol. 81, no. 3, pp. 317–330, 2009.

[20] P. Bradley and O. Mangasarian, "k-plane clustering," *Journal of Global Optimization*, vol. 16, no. 1, pp. 23–32, 2000.

[21] T. T. Zhang, A. Szlam, and G. Lerman, "Median k-flats for hybrid linear modeling with many outliers," *In Workshop on Subspace Methods*, pp. 234–241, 2009.

[22] M. Fischler and R. Bolles, "Ransac random sample consensus: A paradigm for model fitting with ap-plications to image analysis and automated cartography," *Communications of the ACM*, vol. 26, pp. 381–395, 1981.

[23] G. Lerman, M. McCoy, J. Tropp, and T. Zhang, "Robust computation of linear models by convex relaxation," *Foundations of Computational Mathematics*, vol. 15, no. 2, pp. 363–410, 2015.

[24] B. Hartmann, O. Baenfer, O. Nelles, A. Sodja, L. Teslic, and I. Škrjanc, "Supervised hierarhical clustering in fuzzy model identification," *IEEE Trans. on Fuzzy Systems*, vol. 19, no. 6, pp. 1163–1176, 2011.

[25] I. Škrjanc, "Evolving fuzzy-model-based design of experiments with supervised hierarchical clustering," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 861 – 871, 2014.

[26] L. Teslić, B. Hartmann, O. Nelles, and I. Škrjanc, "Nonlinear system identification by gustafsonkessel fuzzy clustering and supervised local model network learning for the drug absorption spectra process," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1941–1951, 2011.

[27] H. R. Singh, S. K. Biswas, and B. Purkayastha, "A neuro-fuzzy classification technique using dynamic clustering and gss rule generation," *Journal of Computational and Applied Mathematics*, vol. 309, no. 1, pp. 683 – 694, 2017.

[28] I. Škrjanc, S. Ozawa, T. Ban, and D. Dovžan, "Large-scale cyber attacks monitoring using evolving cauchy possibilistic clustering," *Applied Soft Computing*, vol. 62, pp. 2833–2839, 2017.

[29] I. Škrjanc, S. Blažič, E. Lughofer, and D. Dovžan, "Inner matrix norms in evolving cauchy possibilistic clustering for classification and regression from data streams," *Information Sciences*, vol. 478, 2018.

[30] M. C. Tsakiris and R. Vidal, "Hyperplane clustering via dual principal component pursuit," *arXiv:1706.01604v2*, pp. 1–48, 2017.

[31] O. Nelles, S. Sinsel, and R. Isermann, "Local basis function networks for identification of turbocharger," in *IEEE UKACC Int. Conf Control*, 1993, pp. 7–12.

[32] O. Banfer, B. Hartmann, and O. Nelles, "Polymot versus hilomot - a comparison of two different training algorithms for local model networks," in *16th IFAC Symposium on System Identification*, July 2012, pp. 1569–1574.

[33] B. Hartmann, T. Ebert, T. Fischer, J. Belz, G. Kampmann, and O. Nelles, "Lmntool - toolbox zum automatischen trainieren lokaler modellnetze," in *22. Workshop Computational Intelligence*, 2012, p. 1.

[34] J. Abonyi, *Fuzzy Model Identification for Control*. Boston: Birkhuser, 2003.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] J.Anonimus, "Fuzzy model identification toolbox," *https://will.be.added.later*, pp. 1–2, 2017.

[37] J. H. Friedman, "Multivariate adaptive regression splines (with discussion)," *Ann. Statist.*, vol. 19, no. 1, pp. 1–141, 1991.

[38] M. C. Mackey and L. Glass, "Oscillations and chaos in physiological control systems," *Science*, vol. 9, pp. 197–287, 1977.

[39] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in *IEEE European Control Conference 2013*, 2013, pp. 2933–2938.

[40] D. Aleksovski, D. Dovžan, S. Džeroski, and J. Kocijan, "A comparison of fuzzy identification methods on benchmark datasets," *IFAC-PapersOnLine*, vol. 49, no. 5, pp. 31 – 36, 2016.

Dejan Dovžan received the B.Sc. degree from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2008. In his bachelor thesis, he covered self-tuning algorithms for a PFC controller and proposed solutions for better disturbance rejection of the PFC. In 2013, he received the Ph.D. degree also from the University of Ljubljana.

In his Ph.D. degree, he was covering the aspects of online and evolving methods for control and modeling purposes. He is currently an Assistant Professor with the Laboratory of Autonomous Mobile Systems, University of Ljubljana. His main research interests include fuzzy model learning methods, recursive and evolving fuzzy model identification, modeling from data, and model-predictive control.

Dr. Dovžan received the Preeren Award for his bachelor thesis. For his Ph.D. thesis, he received the Vodovnik award for an exceptional Ph.D.

Igor Škrjanc received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical and Computer Engineering, University of Ljubljana, Ljubljana, Slovenia, in 1988, 1991, and 1996, respectively.

He is currently a Professor of Automatic Control with the Faculty of Electrical Engineering, University of Ljubljana, and the head of the research program Modeling, Simulation and Control. His main research interests include intelligent, predictive control systems, and autonomous mobile systems.

Dr. Škrjanc received the Highest Research Award from the Faculty of Electrical Engineering, University of Ljubljana, in 2007, and the Highest Award of the Republic of Slovenia for Scientific and Research Achievements in 2008. He also received the Zois Award for outstanding research results in the field of intelligent control. He also received the Humboldt Research Fellowship for Experienced Researchers for the period between 2009 and 2011.