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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

Incremental Fuzzy C-regression Clustering from Streaming Data for Local-model-network Identification

Sašo Blažič, Igor Škrjanc *Member, IEEE*

***Abstract*—In this paper, a new approach to evolving fuzzy model identification from streaming data is given. The structure of the model is given as a local-model network in Takagi-Sugeno form, and the partitioning of the input-output space is based on metrics in which these local models are defined as prototypes of the clusters. This means that the clusters and the local models share the same parameters; therefore, the number of parameters of the evolving system is much lower in comparison to similar systems of comparable complexity, and the problems of parameter identifiability are not a particular issue. The algorithm adds the local models in an incremental fashion and recursively adapts the local model parameters. The proposed algorithm was tested on three examples to demonstrate the main features. The first example is a simple simulated example with intersecting clusters; the second is a very-well known benchmark that treats the Mackey-Glass time series; the third is an example that shows the classification of the data from a laser rangefinder. These examples show the great potential of the proposed approach in certain applications.**

***Index Terms*—Incremental clustering, Fuzzy C-regression clustering, Stream data, Evolving fuzzy model identification.**

I. Introduction

In recent years, the processing of streaming data has become increasingly important. This is because currently the data are generated continuously by thousands of data sources, and include not only the data generated in factories and processing facilities, and traffic control, but also a wide variety of data including log files generated by customers using mobile or web applications, e-commerce purchases, financial trading, information from social networks, geo-spatial services, telemetry, etc. Such data need to be processed sequentially and incrementally on a record-by-record basis or over sliding time windows to obtain the information about the behavior instantaneously. This has sparked an increased interest in the on-line identification of nonlinear models that combine fuzzy logic and neuro-fuzzy networks, as presented, for example, in [1]–[13]. Essentially, these methods are based on various fuzzy clustering algorithms, but they are modified and extended for processing the data streams. The extension of the Gustafson-Kessel clustering algorithm for data stream clustering is presented in [14] and [15], in [16] the recursive method based on the Gath-Geva clustering algorithm is given, while an evolving clustering method (ECM) is proposed in [2]. Much attention has been given to an evolving algorithm that adapts the structure of Takagi-Sugeno fuzzy model (eTS) [1]. The identification of a fuzzy model in general requires the partitioning of the input-output space in the first phase and the estimation of local-linear parameters in the second phase. The partitioning can be done either using some a priori knowledge or, more efficiently, by implementing a clustering algorithm. Combining the latter with a changeable model structure results in an evolving paradigm, in which the rule base of the fuzzy model is updated when a new data sample from the stream is available; the estimation of the clusters parameters is calculated by the recursive clustering algorithm [17] in which subtractive clustering is used as presented in [18], [19]. The self-tuning of membership functions in which the parameters are adjusted automatically is presented in an extended Takagi-Sugeno model (exTS [20] and in eTS+ [21]). This enables the detection of clusters of various shapes. The algorithms differ from the algorithm proposed in [1] regarding the calculation of the fuzzy covariance matrix and in the adaptation of cluster centers.

In all the above-mentioned algorithms, the clustering is based on the point-shaped or hyper-spherical-shaped partitioning of the data space. A very different approach is presented in [22], in which a modified version of fuzzy c-means clustering was proposed, called the fuzzy c-regression clustering (FCRM) algorithm, which develops the clusters in the form of hyper-planes. The FCRM algorithm groups the input-output data into c clusters defined with c hyper-planes. These hyper-planes represent both the premise and the consequence parts of the rules in the identified Takagi-Sugeno model. This means that the hyper-planes in the premise and in the consequence part have the same parameters, which are then updated at the same time with the incoming streaming data, [11]. The proposed algorithm has two major problems: the sensitivity to noise [23], [24] and the initialization problem, which may cause the algorithm to converge to a local minimum of the objective function [25]. The introduction of various objective functions for a robust version of FCRM has been successfully implemented and reported in [24], [26], and [27]. The FCRM algorithm was applied to construct the fuzzy model in [28] and [29]. In [30], a new FCRM clustering algorithm (NFCRMA) is presented, which is deduced from the fuzzy clustering objective function of FCRM with the Lagrange multiplier rule, possessing an integrative and concise structure. The proposed approach primarily consists of two steps: premise parameters identification and consequent parameters identification. Some authors attempt to employ an evolutionary computational technique (particle swarm optimization (PSO)) to estimate the

S. Blažič and I. Škrjanc are with University of Ljubljana, Faculty of Electrical Engineering, Ljubljana, Slovenia.

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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

parameters of the consequence part, such as proposed in [26] and [31]. In [32], an affine Takagi-Sugeno fuzzy modelling algorithm by fuzzy c-regression model clustering is proposed, which involves a novel cluster validity criterion to set up the appropriate number of clusters. Fuzzy membership functions are the crucial element that mainly affects the model structure and modeling accuracy. In [33], a new hyperplane-shaped fuzzy membership function is designed to match the main requirements for T-S fuzzy model identification.

In this paper, we attempt to combine the advantages of the FCRM: specifically, the more compact and suitable structure for the design of Takagi-Sugeno model and the much lower number of parameters to be estimated, together with the evolving nature of the algorithms described before. This means that the algorithm adds a new local model when necessary, specifically if the measured data sample from the data stream does not belong to the existing local models. The algorithm then increments the number of local models and initializes the parameters of that model in an on-line manner. The salient features of the proposed algorithm are: 1) low number of design parameters together with low sensitivity of operation with respect to their value (which makes the tuning very simple for the user); 2) low number of identified parameters (thereby making the algorithm more robust and less sensitive to noise in comparison with methods of similar complexity); 3) simple one-parameter control over modelling error vs. model complexity trade-off; 4) ability to handle linear and nonlinear systems while keeping the model transparent by adopting affine local models.

The paper is organized as follows. In Section II, the nonlinear model described as local model network is briefly introduced. In Section III, the incremental c-regression and c-varieties clustering methods are presented in detail; how to add a local model and how to adapt the current model parameters are explained. In Section IV, the model prediction and model simulation based on the proposed model with affine prototypes are explained. In Section V, the proposed algorithm was tested on three examples to show the main features.

II. Nonlinear model described as local model network

A nonlinear system is often represented by an appropriately weighted sum of local models of certain type. Here, we will assume that the output of the system y can be modelled using a local model network proposed in [34] given by

$$y = \sum_{j=1}^{m} \mu_j(\boldsymbol{u}_p) y_j(\boldsymbol{u}) \tag{1}$$

where $\boldsymbol{u}_p \in \mathbb{R}^q$, $\boldsymbol{u} \in \mathbb{R}^r$, $\mu_j \in [0,1]$ gives validity of the respective local model output y_j, i.e. $\mu_j(\boldsymbol{u}_p)$ defines the regions in the space of $\boldsymbol{u}_p$ where the local models are valid. The validity function μ_j is constructed so that the partition of unity is fulfilled in the convex set $\mathcal{C}$ that includes the whole region of interest of $\boldsymbol{u}_p$:

$$\sum_{j=1}^{m} \mu_j(\boldsymbol{u}_p) = 1 \quad \forall \boldsymbol{u}_p \in \mathcal{C} \subseteq \mathbb{R}^q \tag{2}$$

In the context of Takagi-Sugeno fuzzy models, q-element vector $\boldsymbol{u}_p$ represents antecedent variables, r-element vector $\boldsymbol{u}$ represents consequent variables, $\mu_j(\cdot)$ define membership functions, and m is the number of rules in the rule base. Eq. (1) can be seen as a mapping from $\boldsymbol{u}_p$ and $\boldsymbol{u}$ to y. According to the nature of the system, the variables in $\boldsymbol{u}_p^T = [u_{p1}, \ldots, u_{pq}]$ that define the partitioning of the input-output space are not necessarily the inputs included in the regression vector $\boldsymbol{u}$.

The functions $y_j(\boldsymbol{u})$ can take an almost arbitrary form although linear or affine functions are often used for the sake of simplicity. In our case, affine functions will be utilized:

$$y_j(\boldsymbol{u}) = \theta_{j0} + \theta_{j1}u_1 + \theta_{j2}u_2 + \ldots + \theta_{j,r}u_r \quad j = 1, \ldots, m \tag{3}$$

where the r-element vector $\boldsymbol{u}^T = [u_1, u_2, \ldots, u_r]$ has been introduced. To simplify further derivations, we shall also use the augmented vector $\boldsymbol{u}_e^T = [1, u_1, u_2, \ldots, u_r]$ and the corresponding vector of parameters $\boldsymbol{\theta}_j^T = [\theta_{j0}, \theta_{j1}, \ldots, \theta_{j,r}]$. Combining Eqs. (1) and (3) we obtain

$$y = \sum_{j=1}^{m} \mu_j(\boldsymbol{u}_p) \boldsymbol{u}_e^T \boldsymbol{\theta}_j = \sum_{j=1}^{m} \boldsymbol{\psi}_j^T \boldsymbol{\theta}_j \tag{4}$$

where the weighted augmented regression vector

$$\boldsymbol{\psi}_j^T = \mu_j(\boldsymbol{u}_p) \boldsymbol{u}_e^T \quad j = 1, \ldots, m \tag{5}$$

has been introduced.

III. Model identification using affine prototypes

One of the main issues when working with local model networks is obtaining the validities of the local models based on the current set of data stored in the vector $\boldsymbol{u}_p$. Very often, some prototype is defined for the characteristic data that represent the data in the j-th local model, and the validity of this local model is obtained as a function of the distance with respect to the prototype. Usually, the prototype of the data representing the j-th local model is a point in $\mathbb{R}^q$. while there are also other parameters that have to be known to calculate the validity of the model exactly.

In contrast, data clustering is often performed in the space of a higher dimension $\mathbb{R}^{q+1}$, where measurements $\boldsymbol{z}^T = \left[\boldsymbol{u}_p^T, y\right]$ are depicted. This is the path also taken in this paper. Here, linear (or better affine) functions will be used to represent the prototypes of the clusters instead of points, which are usually used to mark cluster centers. This means that the data belonging to a certain local model (often also referred to as a cluster) will be represented by a manifold in the problem space $\mathbb{R}^{q+1}$. The dimension of the prototype is defined by parameter s, where $0 \leq s \leq q$. If the cluster prototypes are represented by points, the dimension of the manifold is $s = 0$: similarly, $s = 1$ for straight lines, $s = 2$ for planes, and $s > 2$ for subspaces of higher dimensions. In this paper, we will assume that the prototypes are hyperplanes, which means that their dimension is q or, equivalently, the prototypes that lie inside the problem space $\mathbb{R}^{q+1}$ have the codimension of 1.

A unique property of the proposed approach is that the affine functions originally used for the affine local model given by Eq. (3) also define the affine prototypes of the clusters in the space of $\boldsymbol{z}$. Consequently, the local affine model

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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

and the representation of the affine prototype share the same parameters. Obviously, the original description of the system model given by Eq. (1) also slightly modifies to

$$y = \sum_{j=1}^{m} \mu_j(\boldsymbol{u}, y) y_j(\boldsymbol{u}) = \sum_{j=1}^{m} \mu_j(\boldsymbol{u}, y) \boldsymbol{u}_e^T \boldsymbol{\theta}_j \tag{6}$$

and the dimension q becomes r. Therefore, only r will be used in the remainder of the paper. Note that not only the input $\boldsymbol{u}$ but also the output y is used when calculating the distance from the prototype hyperplane, meaning that the output is needed when validity μ_j is being calculated.

The validity of local models is calculated based on the distance from the current data to the affine prototypes. The distance is then used as proposed in fuzzy c-means clustering algorithm [35]. It is based on the minimization of c-means objective function. Solving the minimization problem leads to the validities of the respective local models at time instant k:

$$\mu_{jk} = \left(d_{jk}^2 \sum_{i=1}^{m} \left(\frac{1}{d_{ik}^2} \right)^{\frac{1}{\eta - 1}} \right)^{-1} \quad j = 1, \ldots, m \tag{7}$$

where d_{jk}^2 defines the distance between the measurement at time k

$$\boldsymbol{z}_k = \left[\boldsymbol{u}_k^T, y_k \right]^T = \left[u_{1k}, \ldots, u_{rk}, y_k \right]^T \tag{8}$$

and the j-th local prototype, and η is a design parameter that describes the interpolation between the prototypes (the value of 2 usually gives good results). It is simple to show that the local model validities fulfil the conditions for the partition of unity

$$\mu_{jk} \in [0, 1] \quad \forall j, k \tag{9}$$

$$\sum_{j=1}^{m} \mu_{jk} = 1 \quad \forall k \tag{10}$$

In this paper, two different ways of calculating the distance d_{jk}^2 will be proposed. Consequently, two versions of the model identification algorithm will be obtained. In the first case, d_{jk}^2 is simply the square of the difference between the current system output y_k and the prediction based on the j-th local affine function:

$$d_{jk}^2 = \left(y_k - \boldsymbol{u}_{ek}^T \boldsymbol{\theta}_j \right)^2 \quad j = 1, \ldots, m \tag{11}$$

In the second version of the algorithm, the orthogonal distance between the data point $\boldsymbol{z}_k$ from Eq. (8) and the j-th cluster prototype is used. The orthogonal distance is the projection of the vector $(\boldsymbol{z}_k - \boldsymbol{v}_j)$ to the normal vector of the j-th prototype defined as $\boldsymbol{h}_j$, where $\boldsymbol{v}_j$ stands for the center or any other point on the j-th cluster prototype. The projection can be calculated using a scalar product as follows

$$d_{jk}^2 = \frac{\left(\left(\boldsymbol{z}_k - \boldsymbol{v}_j \right)^T \cdot \boldsymbol{h}_j \right)^2}{\boldsymbol{h}_j^T \boldsymbol{h}_j} \quad j = 1, \ldots, m \tag{12}$$

By taking into account the fact that the orthogonal vector to the j-th cluster prototype is described by using the local model parameters as $\boldsymbol{h}_j = [-\theta_{j1}, \ldots, -\theta_{jr}, 1]^T$ and $\boldsymbol{v}_j^T \cdot \boldsymbol{h}_j = \theta_{j0}$ which follows from Eq. (3), the following description of the distance is obtained

$$d_{jk}^2 = \frac{\left(y_{jk} - \boldsymbol{u}_{ek}^T \boldsymbol{\theta}_j \right)^2}{1 + \sum_{s=1}^{r} \theta_{js}^2} \quad j = 1, \ldots, m \tag{13}$$

Any of the distances given by Eqs. (11) or (13) can be used to calculate the distances of the current measurement $\boldsymbol{z}_k$ to the existing affine prototypes, which in turn gives the validities of the local models defined by Eq. (7). Thus, two variants of model identification for the estimation of system parameters can be derived. In both cases, either an offline or an online algorithm can be implemented.

When dealing with data streams, an online method that continuously updates model parameters needs to be implemented. In our case, a weighted recursive least square algorithm is adopted, which consists of the following steps [36]:

1) The weighted estimation error e_{jk} at time instant k, the error between the current weighted output $y_{jk} = \mu_{jk} y_k$ and the weighted model output based on old parameter estimate $\boldsymbol{\psi}_{jk}^T \boldsymbol{\theta}_{j,k-1}$, is calculated as follows

$$e_{jk} = y_{jk} - \boldsymbol{\psi}_{jk}^T \boldsymbol{\theta}_{j,k-1} = \mu_{jk} \left(y_k - \boldsymbol{u}_{ek}^T \boldsymbol{\theta}_{j,k-1} \right) \quad j = 1, \ldots, m \tag{14}$$

2) The innovation gain vector $\boldsymbol{K}_{jk}$ at time instant k is given in the following way

$$\boldsymbol{K}_{jk} = \boldsymbol{P}_{j,k-1} \boldsymbol{\psi}_{jk} \left(\gamma + \boldsymbol{\psi}_{jk}^T \boldsymbol{P}_{j,k-1} \boldsymbol{\psi}_{jk} \right)^{-1} \quad j = 1, \ldots, m \tag{15}$$

where $\boldsymbol{P}_{j,k-1}$ is the estimate-error covariance matrix at time instant $(k-1)$, and $0 < \gamma \leq 1$ stands for the forgetting factor, which is to be selected by the user (this value is usually between 0.95 and 1).

3) The estimate-error covariance matrix $\boldsymbol{P}_{jk}$ is calculated as

$$\boldsymbol{P}_{jk} = \frac{1}{\gamma} \left(\mathbf{I} - \boldsymbol{K}_{jk} \boldsymbol{\psi}_{jk}^T \right) \boldsymbol{P}_{j,k-1} \quad j = 1, \ldots, m \tag{16}$$

4) The current model parameters $\boldsymbol{\theta}_j$ are updated:

$$\boldsymbol{\theta}_{jk} = \boldsymbol{\theta}_{j,k-1} + \boldsymbol{K}_{jk} e_{jk} \quad j = 1, \ldots, m \tag{17}$$

Note that the steps of the above algorithm refer to the parameters of the j-th local model. Various versions of the algorithm make different choices regarding which local models to update at a certain time instant (either all local models or only the winning-cluster local model) or the local clusters with validity exceeding some predefined threshold. A general rule is that only models with sufficient excitation should be updated if problems such as parameter drift are to be avoided. The analysis of the above algorithm shows that the update of the models with low validity is very small due to the quadratic dependence on μ_{jk}, which can be seen by introducing Eqs. (14) and (15) into (17) and taking into account Eq. (5):

$$\boldsymbol{\theta}_{jk} = \boldsymbol{\theta}_{j,k-1} + \frac{\mu_{jk}^2 \left(y_k - \boldsymbol{u}_{ek}^T \boldsymbol{\theta}_{j,k-1} \right)}{\gamma + \boldsymbol{\psi}_{jk}^T \boldsymbol{P}_{j,k-1} \boldsymbol{\psi}_{jk}} \boldsymbol{P}_{j,k-1} \boldsymbol{u}_{ek} \tag{18}$$

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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

Note that the algorithm given in this paper is not confined to the choice of the online parameter-estimation method presented here. One can rely on some other recursive-least-squares-based algorithm for fuzzy models or even apply an algorithm utilizing a different cost function.

This on-line method is not an incremental one, i.e. the number of local models is kept fixed regardless of the streaming data. For this reason, the method is very sensitive to the initialization. The inherent property of the algorithm is that the size of a prototype is infinite; therefore, the algorithm tends to merge collinear clusters, despite the fact that the actual clusters are well separated. The algorithm can also converge towards false partitions if the clusters have significantly different volumes.

IV. Incremental c-regression and c-varieties clustering method

The algorithm presented in Section III enables the identification of the local-model network given by Eq. (6). Its main drawback is that it requires the knowledge of the number of clusters, which should be defined in advance. It is also difficult to cope with systems in which the conditions change, since the structure of the model is fixed. For this reason, two variants of incremental models are proposed. These algorithms can be seen as an incremental c-regression and an incremental c-varieties clustering algorithm. They only differ in the calculation of the distance between new measurements and existing local model prototypes.

The proposed incremental algorithm starts completely from scratch, without any data and any local models. When the data stream starts, measurements usually arrive at a certain sample rate. After the arrival of a new measurement $\boldsymbol{z}_k$, one on the following actions has to be taken:

- If the measurement lies "close" to an existing local model prototype, it is decided that the measurement belongs to the corresponding cluster. Consequently, the parameters of the corresponding local model are updated following the procedure given in Section III.
- If the measurement is "far" from the existing clusters but not "very far" (meaning that it is not regarded as an outlier), the data are put in the buffer. The content of the buffer is analyzed after each update of the buffer in order to either use this data to construct a new local model or simply discard all these data if the informational content is not adequate.
- If the measurement lies "very far" from the existing clusters, it is decided that it is probably an outlier, and the data are not stored anywhere.

In the above items, soft terms are used for specifying the size of the distance with respect to the existing clusters. The meaning of "close", "far", etc. will be quantified based on one of the distances d_{jk}^2 given by Eqs. (11) or (13). When associating a measurement at time k with a j-th cluster, we take into account the distance d_{jk}^2 between the measurement and the local model hyperplane at this particular time. The winning cluster is the one with the lowest distance; its index will be denoted with w ($1 \leq w \leq m$)

$$w = \arg \min_{1 \leq j \leq m} d_{jk}^2 \tag{19}$$

The mean value of the past distances $\overline{d_j^2}$ is kept for each cluster. It is updated recursively upon the addition of a new measurement to the winning cluster. At time k, only $\overline{d_j^2}$ of the winning cluster is updated, as proposed in [37]:

$$\overline{d^2}_{wk} = \frac{k_w - 1}{k_w} \overline{d^2}_{w,k-1} + \frac{1}{k_w} d_{wk}^2 \tag{20}$$

where k_j stands for the current number of samples in the j-th cluster (k_w is the number of samples in the winning cluster), and index of $\overline{d_w^2}$ is the current time. Initialization at time $t = 0$: $\overline{d^2}_{j,0} = 0$ for all clusters.

A. Keeping unclassified data in the buffer

Probably the most crucial part of an incremental system is the decision regarding how and when a new local model (or cluster) is to be added. When a new measurement $\boldsymbol{z}_k$ arrives, the distances $d_{jk}^2 (j = 1, 2, \ldots, m)$ from existing local models are calculated. If this distance is considerably higher than the mean of the previous distances of the data in the cluster, i.e.,

$$d_{wk}^2 > \kappa_{min} \overline{d^2}_{w,k-1} \tag{21}$$

the decision is made that this particular point (or piece of data) will not be associated with any of the existing clusters. Although being a relative (dimensionless) quantity by its definition, Eq. (21), κ_{min} still has to be tuned in order to achieve better results. This can be done offline on a smaller batch of data. We have tested the proposed approach on many different data sets, and good results are usually obtained if κ_{min} is in the range of 1 to 100. Note that due to being a relative quantity this parameter cannot be very large or very small, but still it is the only design parameter with significant impact on the performance of the algorithm. In fact, the user can control the trade-off between the achieved modelling error and the number of the local models quite easily by tuning κ_{min}.

However, to prevent false clusters from being introduced based on outliers and also to prevent large transients in the beginning of the cluster lifetime, the data not associated with any local models are not used immediately to initialize a new local model. Rather, they are kept in the buffer. When it is decided that the data in the buffer can form a new local model, a new model is initialized based on the data in the buffer. Several criteria can be implemented for this, but we chose the one that requires that the data in the buffer span a unique hyperplane of dimension r:

$$y = \boldsymbol{u}_e^T \boldsymbol{\theta}_J = \theta_{J0} + \theta_{J1} u_1 + \theta_{J2} u_2 + \ldots + \theta_{J,r} u_r \tag{22}$$

This means that at least $(r + 1)$ measurements are needed in the buffer before this condition is met. Additional criteria can also be applied. For practical purposes, it is not recommended to form a new model based on the nonconsecutive patches of data. Our implementation: if the difference in time index of the consecutive measurements in the buffer exceeds a certain

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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

value (in our case 3), the old batches are flushed while the current batch can be kept.

It has to be noted that the data go through the buffer and the above check also in the case of starting the procedure from scratch. During the starting phase, the data are collected until the first local model can be formed.

B. Adding the prototypes

When a decision is made to construct a new local model, the number of local models m is increased by 1, and all the parameters defining the new local cluster have to be initialized. The parameter vector of the local affine model that also defines the cluster prototype can be obtained from Eq. (22) as $\boldsymbol{\theta}_m = \boldsymbol{\theta}_J$ using the well-known least square method for linear systems. The initialization of other cluster parameters $(\boldsymbol{P}_m, k_m, \overline{d_m^2})$ is also straight-forward.

C. Handling the outliers

It has to be noted that the samples that fulfill condition given by Eq. (21) can lie very far from all the existing clusters. These samples may be outliers and, therefore, the associated data are not used or stored. The condition for classifying a measurement as an outlier is that the following condition similar to Eq. (21) is met:

$$d_{wk}^2 > \kappa_{max}\overline{d^2}_{w,k-1} \tag{23}$$

where $\kappa_{max} > \kappa_{min}$ is a design parameter with the recommended value of $2\kappa_{min}$. If the condition given by Eq. (23) is met, the current sample is not taken into account for adapting the current prototype or adding a new prototype, nor it is stored in the buffer.

The pseudo-code of the incremental fuzzy c-regression and c-varieties algorithm in on-line identification is given in Algorithm 1. Note that in this algorithm all the local models ($j = 1, 2, \ldots, m$) can be updated in Line 16, or just the winning cluster local model ($j = w$). The latter option often produces better results.

V. Model prediction and model simulation based on the proposed model with affine prototypes

The model obtained with the proposed methodology can easily be used for classification in which each new data sample carries both the information on the input vector $\boldsymbol{u}$ and the output y. The classification can, therefore, be performed based on the lowest distance with respect to the affine prototypes.

In the case of model prediction and model simulation, the output y is not known. Actually, it is the goal of model prediction and model simulation to calculate the model output y based on the known model input vector $\boldsymbol{u}$. We will denote this calculated (predicted/simulated) model output $\hat{y}$ to distinguish it from the measured one (denoted y). Simulation or prediction is usually done in a loop running in time. Past measurements $(y_{k-i}, i > 0)$ are included in $\boldsymbol{u}_k$ in the calculation of a predicted output $\hat{y}_k$ while past calculated outputs $(\hat{y}_{k-i}, i > 0)$ are used during the calculation of a simulated output $\hat{y}_k$.

In the case of model prediction and/or model simulation, a problem arises due to the fact that the input data stream only

Algorithm 1 Pseudo-code of the incremental c-regression or c-varieties clustering method in on-line identification

1: Definition of the parameter η, the forgetting factor γ, the parameters κ_{min} and κ_{max}.
2: Initialization of the number of clusters ($m = 0$).
3: **for all** k **do**
4: Computation of the distances $d_{jk},\ j = 1, \ldots, m$. For c-regression method

$$d_{jk}^2 = \left(y_{jk} - \boldsymbol{\psi}_{jk}^T\boldsymbol{\theta}_j\right)^2,\ \ j = 1, \ldots, m$$

and for c-varieties

$$d_{jk}^2 = \frac{\left(y_{jk} - \boldsymbol{\psi}_j^T\boldsymbol{\theta}_j\right)^2}{1 + \sum_{s=1}^{r-1}\theta_{js}^2},\ \ j = 1, \ldots, m$$

5: Determine the cluster with lowest distance (if $m \neq 0$):

$$w = \arg\min_j d_{jk}^2$$

6: **if** $(m == 0)$ **or** $(d_{wk}^2 > \kappa_{min}\overline{d^2}_{w,k-1})$ **and** $(d_{wk}^2 \leq \kappa_{max}\overline{d^2}_{w,k-1})$ **then**
7: Put the current measurement $\boldsymbol{z}_k$ to the buffer.
8: **if** the measurements in the buffer span a unique hyperplane of dimension r **then**
9: Add a new local model (increase m by 1) and initialize local model parameters $\boldsymbol{\theta}_m, \boldsymbol{P}_m, k_m, \overline{d_m^2}$.
10: Empty the buffer.
11: **else if** the difference of time indexes of consecutive measurements in the buffer exceeds 3 **then**
12: Empty the buffer.
13: **end if**
14: **else if** $d_{wk}^2 \leq \kappa_{min}\overline{d^2}_{w,k-1}$ **then**
15: Computation of the new validities for $j = 1, \ldots, m$

$$\mu_{jk} = \left(d_{jk}^2 \sum_{i=1}^{m}\left(\frac{1}{d_{ik}^2}\right)^{\frac{1}{\eta-1}}\right)^{-1}$$

16: Update of local model parameters :

$$e_{jk} = y_{jk} - \boldsymbol{\psi}_{jk}^T\boldsymbol{\theta}_{j,k-1}$$

$$\boldsymbol{K}_{jk} = \boldsymbol{P}_{j,k-1}\boldsymbol{\psi}_{jk}\left(\gamma + \boldsymbol{\psi}_{jk}^T\boldsymbol{P}_{j,k-1}\boldsymbol{\psi}_{jk}\right)^{-1}$$

$$\boldsymbol{P}_{jk} = \frac{1}{\gamma}\left(\mathbf{I} - \boldsymbol{K}_{jk}\boldsymbol{\psi}_{jk}^T\right)\boldsymbol{P}_{j,k-1}$$

$$\boldsymbol{\theta}_{jk} = \boldsymbol{\theta}_{j,k-1} + \boldsymbol{K}_{jk}e_{jk}$$

17: Update of the parameters for the winning cluster:

$$\overline{d^2}_{wk} = \frac{k_w - 1}{k_w}\overline{d^2}_{w,k-1} + \frac{1}{k_w}d_{wk}^2$$

$$\overline{d^2}_{wk} = \overline{d^2}_{w,k-1} \quad j = 1, 2, \ldots, m,\ j \neq w$$

18: **else**
19: The measurement is probably an outlier, and it is disregarded.
20: **end if**
21: **end for**

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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

directly defines the input vector $\boldsymbol{u}$, while the output $\hat{y}$ is to be determined by the model (see Eq. 6)

$$\hat{y} = \boldsymbol{u}_e^T \sum_{j=1}^{m} \mu_j(\boldsymbol{u}, \hat{y}) \boldsymbol{\theta}_j \tag{24}$$

where model validities μ_j $(j = 1, 2, \dots m)$ depend on distances whose calculation need both $\boldsymbol{u}$ and $\hat{y}$. Consequently, direct calculation of $\hat{y}$ is not possible. Some solutions to this problem exist:

- **The use of distance to clusters based on input vectors.** In this approach, the validities of the local models are calculated only taking into account the input vector $\boldsymbol{u}$:

$$\hat{y} = \boldsymbol{u}_e^T \sum_{j=1}^{m} \mu_j(\boldsymbol{u}) \boldsymbol{\theta}_j \tag{25}$$

  Each measurement $\boldsymbol{z}^T = \left[\boldsymbol{u}^T, y\right]$ is associated with a particular (let us assume j-th) local model (or cluster) during the proposed clustering in the training phase (described in Section IV). Only the input-vector part $\boldsymbol{u}$ of these measurements can be collected in separate sets $\mathcal{R}_j$ $(j = 1, 2, \dots m)$ for the purpose of model prediction or model simulation. When, during the prediction/simulation phase, a new input vector $\boldsymbol{u}$ arrives to be mapped to $\hat{y}$, the distance from $\boldsymbol{u}$ to the members of the sets $\mathcal{R}_j$ $(j = 1, 2, \dots c)$ is calculated.
  Note that a very suitable distance measure in this case is a Mahalanobis distance. If each set $\mathcal{R}_j$ of input vectors is described only by its mean $\overline{\boldsymbol{u}}_j$ and its covariance matrix $\boldsymbol{\Sigma}_j^{\boldsymbol{u}}$, the square of the Mahalanobis distance to the current input vector $\boldsymbol{u}_k$ can be computed easily:

$$D_{jk}^2 = (\boldsymbol{u}_k - \overline{\boldsymbol{u}}_j)^T \left(\boldsymbol{\Sigma}_j^{\boldsymbol{u}}\right)^{-1} (\boldsymbol{u}_k - \overline{\boldsymbol{u}}_j) \tag{26}$$

  It is also very simple to recursively adapt the mean and the covariance matrix during the training phase [38].
  The last step before the actual calculation of $\hat{y}$ using Eq. (25) is to map the Mahalanobis distances D_{jk}^2 from Eq. (26) to model validities μ_j. This can be done either using a *winner takes all* principle (the validity of the rule associated with the lowest distance D_{jk}^2 is 1, the others are 0) or mapping D_{jk}^2 to μ_j using the same formula as in Eq. (7).
- **The use of an iterative procedure.** The predicted/simulated output $\hat{y}_k$ can also be obtained iteratively. First, an initial approximation of the predicted output $\hat{y}_k^0$ is chosen. This can be done either by selecting a previous output ($\hat{y}_k^0 = \hat{y}_{k-1}$), or relying on the linear model of the system (given as a single fixed parameter vector $\boldsymbol{\theta}_{lin}$) using $\hat{y}_k^0 = \boldsymbol{u}_e^T \boldsymbol{\theta}_{lin}$. This initial output $\hat{y}_k^0$ is used for obtaining distances to the linear prototypes (using Eqs. 11 or 13) and model validities (using Eq. 7) which in turn result in a model output using Eq. (24). This model output is the next approximation $\hat{y}_k^1$ of the output for which we are searching. It can be used to run a new iteration of the described algorithm. This procedure repeats until the output converges.

There are some remarks concerning the above approaches:

- The first approach is quite simple. While it does not require the elements of the sets $\mathcal{R}_j$ to be stored, the mean values $\overline{\boldsymbol{u}}_j$ and the covariance matrices $\boldsymbol{\Sigma}_j^{\boldsymbol{u}}$ of the elements of the sets $\mathcal{R}_j$ have to be known, but they can be calculated recursively during training (identification phase) [38]. It has to be emphasized that this approach uses affine prototypes of infinite size during the identification phase and finite compact clusters during prediction/simulation phase.
- The second approach relies on an iterative procedure whose convergence needs to be analyzed. Model validities fulfill the conditions for partition of unity (Eqs. 9 and 10), and it is easy to show that $\hat{y}$ in Eq. (24) is always upper-bounded and lower-bounded at the fixed value of $\boldsymbol{u}_e$:

$$\min_{j \in \mathcal{I}_m} \left(\boldsymbol{u}_e^T \boldsymbol{\theta}_j\right) \leq \boldsymbol{u}_e^T \sum_{j=1}^{m} \mu_j(\boldsymbol{u}, \hat{y}) \boldsymbol{\theta}_j \leq \max_{j \in \mathcal{I}_m} \left(\boldsymbol{u}_e^T \boldsymbol{\theta}_j\right) \tag{27}$$

  where $\mathcal{I}_m = \{1, 2, \dots, m\}$ is a set containing the first m integers.
  It is also easy to check that Eq. (24) holds for $\hat{y} = \boldsymbol{u}_e^T \boldsymbol{\theta}_j$ and any j from $\mathcal{I}_m$. This means that all these values represent the equilibria for $\hat{y}$ of the iterative algorithm.
  The boundedness of $\hat{y}$ and existence of equilibria for $\hat{y}$ do not guarantee that the above iterative procedure results in a convergent series, but the guaranteed boundedness of $\hat{y}$ is sufficient for our needs.

VI. Examples

The proposed algorithm was tested on three examples to demonstrate the main features. The first example is a simple simulated example with intersecting clusters, the second is a benchmark that treats the Mackey-Glass time series, and the third is an example that shows the classification of the data from a real laser rangefinder.

One of the most attractive features of the proposed approach is the low number of design parameters; the approach is also quite insensitive to most of parameters. In all our experiments, the parameters η was fixed to 2. The system is also very robust with regards to the forgetting factor, which was kept at $\gamma = 0.98$ during all tests. The only parameter that was important was κ_{min}, while κ_{max} was always set to $2\kappa_{min}$.

A. Simple example with intersecting clusters

This example treats a very simple simulated problem in which there are two processes running in parallel. The first process is described by

$$y_t = x_t + n_t \tag{28}$$

and the second by

$$y_t = -x_t + n_t \tag{29}$$

where x_t is the input series and y_t the output series while the noise n_t is a Gaussian noise with zero mean and the standard deviation of 0.05. Moreover, some outliers are added to the data (corrupting 3% of the data). The data stream consists of the (x, y) pairs, for which the input x is generated randomly

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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

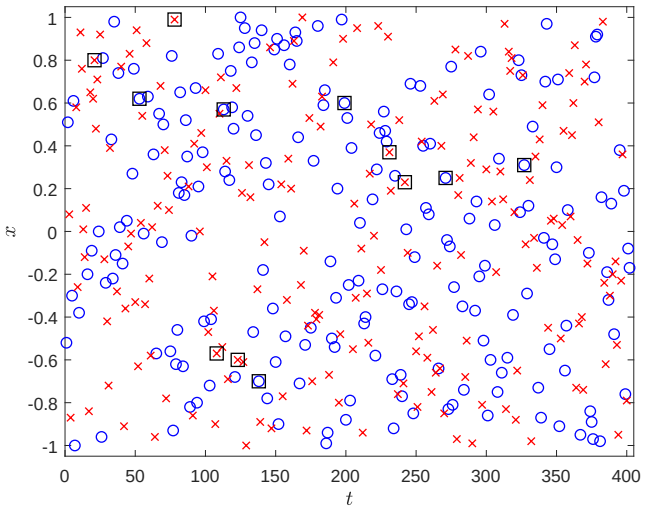

Fig. 1. The data stream: blue circles indicate the samples from process 1, given by Eq. (28); the red crosses indicate the samples generated by process 2 given by Eq. (29)

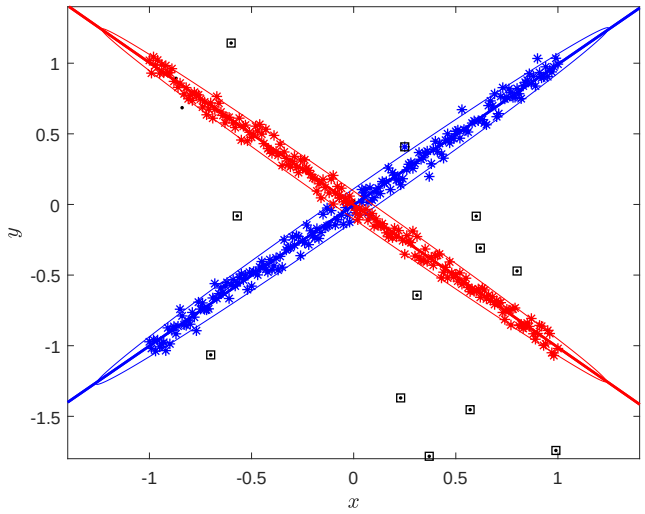

Fig. 2. The clusters obtained in the simple example with intersecting clusters: the two clusters are shown with blue and red; also shown: affine prototypes with thick lines and ellipses with 90% probability

on the $[-1, 1]$ interval; then one of the processes given by Eqs. (28) and (29) is randomly chosen to obtain the output y. The data stream is depicted in Fig. 1. The outliers are surrounded by squares. It is shown that at each particular time t a sample is chosen randomly from one of the two processes with the randomly generated input x. If this data stream is treated by the proposed algorithm ($\kappa_{min} = 10$), two cluster prototypes are obtained as depicted in Fig. 2. The affine prototypes at the end of the experiments are shown by thick lines, the color of the samples shows the corresponding cluster, and ellipses show the cluster area with the 90 % probability. Black dots show the measurements not associated with any of the clusters, and the measurements surrounded by squares depict the outliers. We can see that all the outliers far from the clusters have been treated correctly (they were not associated with any of the clusters). If a measurement has a lot of noise or an outlier is close to a cluster, it is hard for the algorithm to make a clear distinction. In conclusion, we can say that the algorithm is able to cope with measurements generated by different processes running in parallel and that the algorithm is robust to outliers.

B. Mackey-Glass time series

The Mackey-Glass time series is a chaotic time series generated by the differential equation:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \quad x(0) = 1.2, \tau = 17 \quad (30)$$

This continuous nonlinear system has quite often been simulated and sampled in the literature and has become a certain benchmark in the area of evolving systems. In this example, we have been working with a dataset that was created previously and worked on by many researchers. Therefore, we used the same problem setting as in [2]:

- the input vector $\boldsymbol{u}$ includes the samples x_{t-18}, x_{t-12}, x_{t-6}, and x_t;
- the output of the model is the 85-steps-ahead prediction of the output: $y_t = x_{t+85}$;
- 3000 data points ($t \in [201, 3200]$) were used for the training/identification;
- 500 data points ($t \in [5001, 5500]$) were used for the prediction.

Method	Rules	RMSE	$\frac{\text{Rules}}{\text{mean}}$	$\frac{\text{RMSE}}{\text{mean}}$	Sum
DENFIS [2]	58	0.0628	1.4545	1.0248	2.4793
DENFIS [2]	27	0.0920	0.6771	1.5013	2.1784
exTS [39]	10	0.0754	0.2508	1.2304	1.4812
eTS+ [37]	10	0.0892	0.2508	1.4556	1.7064
eTS [1]	113	0.0217	2.8339	0.3541	3.1880
rGK [40]	58	0.0481	1.4545	0.7849	2.2395
rGK [40]	10	0.0862	0.2508	1.4066	1.6574
rFCM [41]	10	0.1039	0.2508	1.6955	1.9462
rFCM [41]	58	0.0702	1.4545	1.1455	2.6001
rFCM [41]	100	0.0285	2.5078	0.4651	2.9729
eFuMo [40]	21	0.0753	0.5266	1.2288	1.7554
eFuMo [40]	41	0.0316	1.0282	0.5157	1.5439
eFuMo [40]	68	0.0224	1.7053	0.3655	2.0709
InFuR ($\kappa_{min} = 50$)	**22**	**0.0392**	**0.5517**	**0.6397**	**1.1914**
InFuR ($\kappa_{min} = 80$)	**18**	**0.0583**	**0.4514**	**0.9514**	**1.4028**
InFuR ($\kappa_{min} = 100$)	**14**	**0.0757**	**0.3511**	**1.2353**	**1.5864**

TABLE I
RESULTS FOR MACKEY-GLASS TIME SERIES.

The results from other methods are taken from [1], [2], [39], [37], [40] and [41], and are shown in Table I. The proposed method InFuR was parameterized as follows: $\eta = 2$, $\gamma = 0.98$, $\kappa_{min} \in \{50, 80, 100\}$, and $\kappa_{max} = 2\kappa_{min}$.

It can be seen that for the Mackey-Glass time-series the presented method InFuR gives results that are very comparable to the other methods, especially to those with a reasonable number of clusters. The best results are obtained with the eTS method according to which, together with 113 local models, the RMSE is equal 0.0217. With the InFuR method, the RMSE of 0.0392 is obtained with only 22 local models. When κ_{min} increases to 80, the number of rules decreases to 18 while RMSE increases to 0.0583. Increasing κ_{min} further to 100, results in a lower number of rules (14) while

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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

RMSE also increases (0.0757). Taking into account only the examples from the literature with similar numbers of local model (DENFIS, 27, rGK, 10, rFCM, 10, eFuMo, 21), the best RMSE is 0.0753. The proposed method shows the best performance with 22 local models together with RMSE of 0.0392. In Table I, the results are also shown in a relative manner, according to which the number of rules was divided by the mean value of rules in all the examples (39.9), presented in the Rules/mean column, and the value of RMSE divided by mean value of RMSE values (0.0613), as RMSE/mean in next column. The sum of both values is given in the last column, denoted as Sum. This result indicates that the proposed approach shows a very good trade-off between the model error and model complexity.

C. Laser rangefinder measurements

The laser rangefinder (LRF) is a sensor that uses a laser beam to determine the distance to an object. LRFs are often used in mobile robotics, primarily for localization purposes, map building, or simultaneous localization and mapping (SLAM) [42]. The robot's pose can be estimated by fusing the data from the LRF and the information from the known map of the environment. This is often done using simple geometric features that are extracted from the data obtained from the LRF. The simplest features are straight lines. The first task in the localization or mapping procedure is, therefore, the extraction of existing lines. This task generally requires two phases: the data clustering phase, in which the points forming a line are isolated, and the phase of a line-parameter estimation.

The proposed approach is very suitable for this task, since it puts the points on a straight line to a cluster. The affine prototype directly defines a straight line. In this example, we are working on the data from the LRF that is shown in the raw form in Fig. 3. The measurements are shown with crosses; the connecting line defines the sequence of measurements. The design parameters were chosen as: $\eta = 2$, $\gamma = 0.98$, $\kappa_{min} = 2.5$, and $\kappa_{max} = 2\kappa_{min}$. The resulting clusters after this stream of data has been processed are shown in Fig. 4, in which individual clusters (straight line segments in this case) are shown with different colors. The sequence in the legend shows the sequence of cluster initialization, while the numbers in the legend define the number of measurements in the clusters. Also shown are ellipses corresponding to the data in the clusters (they define the area where the sample belong to a particular cluster with 90% probability). Note also the black dots; the measurements not included in any cluster. The algorithm deleted these data from the buffer.

The proposed algorithm has been compared to the Gustafson-Kessel (GK) clustering algorithm [43], which is a clustering algorithm based on the assumption that the cluster prototype is a point. Since the distances are calculated using the covariance matrix of the cluster, an arbitrary ellipsoidal shape of the cluster can be obtained. When calculating fuzzy memberships, the boundaries of the clusters can take almost arbitrary shapes, which makes this clustering algorithm a powerful one. One drawback of the algorithm is that the number of clusters needs to be known in advance. In our

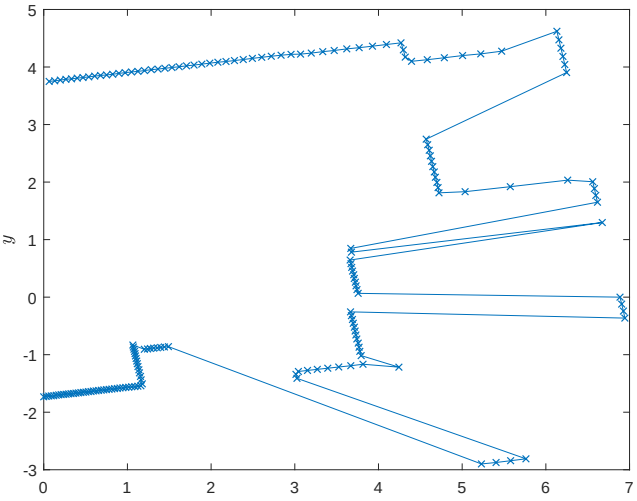

Fig. 3. 2D measurements from the laser rangefinder

case, the algorithm has been run with various numbers of clusters. It turned out that, in the case of a large number of assumed clusters, the algorithm always encounters the problem of the covariance matrix of a cluster becoming singular. Therefore, we did an experiment with the highest number of assumed clusters that produced relevant results. This number was $m = 13$ and the results are shown in Fig. 5. The results in Fig. 5 can be directly compared to the results in Fig. 4. Both algorithms performed quite well but we need to underscore three important shortcomings of the GK algorithm:

- The GK algorithm needs to have the number of clusters available a priori. If a too low number is selected, we face the problem depicted in Fig. 6 in which only $m = 8$ clusters are assumed. Obviously, the algorithm cannot find a very good solution.
- The GK algorithm also suffers from numerical problems that always arise in the case of a too large m, as already mentioned.
- The algorithm is based on random initialisation, which means that different final clusters are in general obtained in different runs of the algorithm.

VII. Conclusion

The proposed approach of clustering in the on-line identification of various processes from the data stream has shown a great potential. The concept of affine prototypes works very well in applications in which the data lie along several segments of different hyperplanes. However, the approach can be used successfully in any application with significant nonlinearity. There are also other benefits of the proposed approach. Since the cluster definition and the local models share the same parameters, the number of parameters of the evolving system is lower in comparison to similar systems of comparable complexity. This means that the problems of parameter identifiability are not a particular issue. Moreover, the number of design parameters that have to be chosen is quite low, and we can mostly work with their "default" values.

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/TFUZZ.2019.2916036, IEEE Transactions on Fuzzy Systems

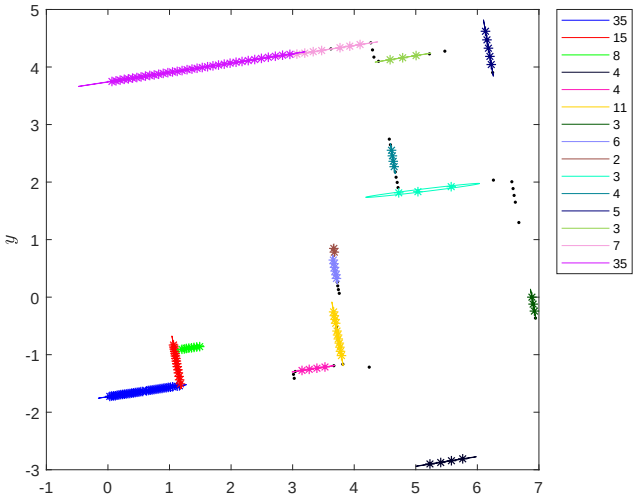

Fig. 4. Clusters obtained after processing the LRF data with the proposed approach

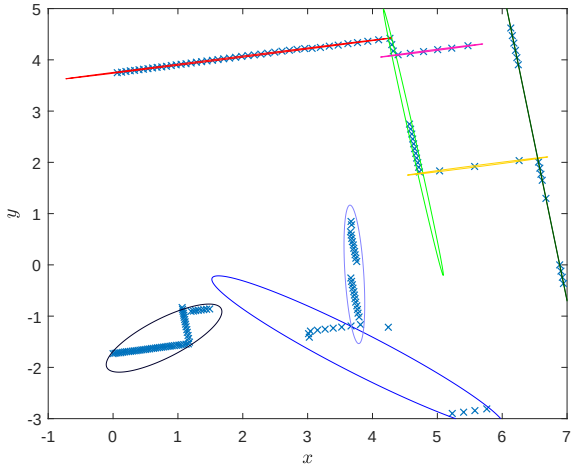

Fig. 6. Clusters obtained after processing the LRF data with the GK algorithm (the number of assumed clusters is 8)

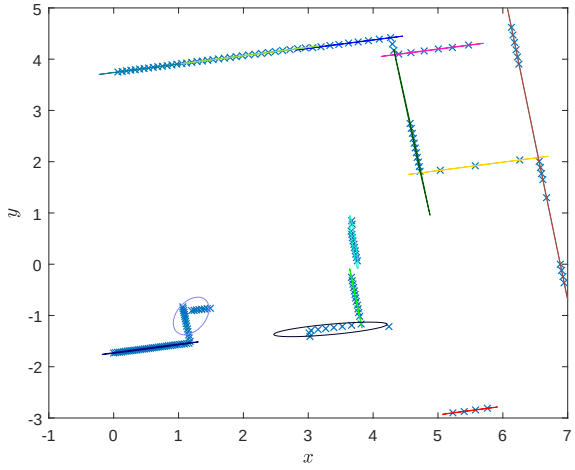

Fig. 5. Clusters obtained after processing the LRF data with the GK algorithm (the number of assumed clusters is 13)

This means that good results are usually obtained without any parameter tweaking, which is certainly something that a practical user would appreciate. The problem of model output prediction is also tackled in this paper. Very often, the validity of local models depends on the current system output, which is not available during simulation/prediction. Two approaches to alleviate this problem are suggested in this paper: the use of distance to clusters based on input vectors, and the use of an iterative procedure.

ACKNOWLEDGEMENT

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

REFERENCES

[1] P. P. Angelov, D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models". *IEEE Trans. Syst. Man Cyber. part B*, vol. 34, no. 1, pp. 484-497, Feb. 2004.

[2] N. K. Kasabov, Q. Song, "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction". *IEEE Trans. on Fuzzy Syst.*, vol. 10, no. 2, pp. 144-154, Apr. 2002.

[3] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning". *IEEE Trans. Syst. Man Cyber. part B*, vol. 31, no. 6, pp. 902-918, Dec. 2001.

[4] N. Kasabov, "Evolving fuzzy neural networks-Algorithms, applications and biological motivation". *Methodologies for the Conceptation, Design and Application of Soft Computing*, Singapore, 1998, pp. 271-274.

[5] Hai-Jun Rong, N. Sundararajan, Guang-Bin Huang, P. Saratchandran, "Sequential Adaptive Fuzzy Inference System (SAFIS) for nonlinear system identification and prediction". *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260-1275, May 2006.

[6] F. J. Lin, C. H. Lin, Po. H. Shen, "Self-Constructing Fuzzy Neural Network Speed Controller for Permanent-Magnet Synchronous Motor Drive". *IEEE Trans. on Fuzzy Syst.*, vol. 9, no. 5, pp. 751-759, Oct. 2001.

[7] S. Wu, M. J. Er, "Dynamic Fuzzy Neural Networks – A Novel Approach to Function Approximation". *IEEE Trans. Syst. Man Cyber. part B*, vol. 30, no. 2, pp. 358-364, Apr. 2000.

[8] S. Wu, M. J. Er, Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamicc fuzzy neural networks". *IEEE Trans. on Fuzzy Syst.*, vol. 9, no. 4, pp. 578-594, Aug. 2001.

[9] C. T. Lin, "A neural fuzzy control system with structure and parameter learning". *Fuzzy Sets and Systems*, vol. 70, pp. 183-212, 1995.

[10] S. G. Tzafestas, K. C. Zikidis, "NeuroFAST: On-Line Neuro-Fuzzy ART-Based Structure and Parameter Learning TSK Model". *IEEE Trans. Syst. Man Cyber. part B*, vol. 31, no. 5, pp. 797-802, Oct. 2001.

[11] K. Kim, J Baek, E. Kim, M. Park, "TSK Fuzzy model based on-line identification", *Proc.* 11^{th} *IFSA World Congress*, Beijing, China, 2005, pp.1435–1439.

[12] M. F. Azeem, M. Hanmandlu, N. Ahmad, "Structure identification of generalized adaptive neuro-fuzzy inference systems". *IEEE Trans. on Fuzzy Syst.*, vol. 11, no. 5, pp. 666-681, Oct. 2003.

[13] I. Škrjanc, J. A. Iglesias Martinez, A. Sanchis, D. Leite, E. Lughofer, F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification : a survey", *Information sciences, In press, 2019.*

[14] D. Filev and O. Georgieva, "An extended version of the Gustafson-Kessel algorithm for evolving data stream clustering", in: Evolving Intelligent Systems: Methodology and Applications, Eds.: P. Angelov, D. Filev, A. Kasabov, John Willey and Sons, IEE Press Series on Computational Intellegence, pp. 273–300, April, 2010.

[15] D. Dovžan and I. Škrjanc, "Recursive clustering based on a Gustafson-Kessel algorithm", Evolving Systems, pp. 15 – 24, vol. 2. 2011.

[16] H. Soleimani-B., C. Lucas and B. N. Araabi, "Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling", Evolving Systems, Springer, vol. 1, issue 1, pp.59–71, 2010.

[17] P. Angelov, "An approach for fuzzy rule-base adaptation using on-line

clustering". Integration of Methods and Hybrid Systems, vol. 35, no. 3, pp. 275–289, 2004.

[18] S. L. Chiu, "Fuzzy model identification based on cluster estimation". Journal of Intellegent and Fuzzy Systems, vol. 2, pp. 267-278, 1994.

[19] R. Yager, and D. Filev, "Generation of fuzzy rules by mountain clustering". J. of Intelligent and Fuzzy Systems 2, pp. 209–219, 1994.

[20] P. Angelov, X. Zhou, Evolving Fuzzy Systems from Data Streams in Real-Time, 2006 International Symposium on Evolving Fuzzy Systems, 7-9 September, 2006, Ambelside, Lake District, UK, IEEE Press, pp.29-35.

[21] P. Angelov, "Evolving Takagi-Sugeno fuzzy systems from streaming data (eTs+)", in: Evolving Intelligent Systems: Methodology and Applications, Eds.: P. Angelov, D. Filev, A. Kasabov, John Willey and Sons, IEE Press Series on Computational Intellegence, pp. 273–300, 2010.

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

[23] J.M. Leski, "Fuzzy c-varieties/elliptotypes clustering in reproducing kernel Hilbert space", Fuzzy Sets Syst., 141, (2), pp. 259–280, 2004.

[24] R.N. Dave, and R. Krishnapuram, "Robust clustering method: A unified view", IEEE Transaction on Fuzzy Systems, 5 (2), pp. 450–465, 1997.

[25] K.C. Ying, S.W. Lin, S.W., Z.J. Lee, and I.L. Lee, "A novel function approximation based on robust fuzzy regression algorithm model and particle swarm optimization", Applied Soft Computing 38(2): pp.1820–1826, 2011.

[26] M. Soltani, A. Chaari, and F. Ben Hmida, "A novel fuzzy c-regression model algorithm using a new error measure and particle swarm optimization" Int. J. Appl. Math. Comput. Sci., Vol. 22, No. 3, 2012.

[27] K. Honda, A. Notsu, and H. Ichihashi, "FPC PCA-guided robust k-means clustering", IEEE Transaction on Pattern Analysis and Machine Intelligence, 18 (1), pp. 67–79, 2010.

[28] E. Kim, M. Park, S. Kim, S. and M. Park, "A transformed input-domain approach to fuzzy modeling", IEEE Transactions on Fuzzy Systems 6(4): pp. 596–604, 1998.

[29] E. Kim, M. Park, S. Ji, and M. Park, "A new approach to fuzzy modeling", IEEE Trans. Fuzzy Syst., 5, (3), pp. 328–337, 1997.

[30] L. Chaoshun, Z. Jianzhong, X. Xiuqiao, L. Qingqing and A. Xueli, "T-S fuzzy model identication based on a novel fuzzy c-regression model clustering algorithm", Engineering Applications of Artificial Intelligence 22 (4-5): pp.646–653, 2009.

[31] N. Qiang, and H. Xinjian, "An improved fuzzy c-means clustering algorithm based on PSO", Journal of Software 6(5): pp. 873–879, 2011.

[32] C.C. Kung and J.Y. Su, "Affine Takagi-Sugeno fuzzy modelling algorithm by fuzzy c-regression models clustering with a novel cluster validity criterion", IET Control Theory Appl., 1, (5), pp. 1255-1265, 2007.

[33] L. Chaoshun, J. Zhou, L. Chang, Z. Huang, and Y. Zhang, "T-S Fuzzy Model Identification Based on a Novel Hyperplane-Shaped Membership Function", IEEE Transactions on Fuzzy Systems 25(5), pp.1364–1370, 2017.

[34] I. Škrjanc, "Fuzzy confidence interval for pH titration curve", Applied Mathematical Modelling, 35(8): pp.4083–4090, 2011.

[35] J.C. Bezdek, "Cluster validity with fuzzy sets", Journal of Cybernetics, 3: pp. 58–73, 1974.

[36] S.S. Haykin, Adaptive filter theory, 4th ed., Prentice Hall, Upper Saddle River, 2002.

[37] P. Angelov, "Evolving Intelligent Systems: Methodology and Applications", New Jersey: Wiley, chapter Evolving Takagi-Sugeno Fuzzy Systems From Streaming Data (eTS+), pp. 21 – 50, 2010.

[38] S. Blažič, P. Angelov and I. Škrjanc, "Comparison of Approaches for Identification of All-data Cloud-based Evolving Systems", IFAC-PapersOnLine, Vol. 48, No. 10, pp. 129–134, 2015. .

[39] P. Angelov and Z. Xiaowei, "Evolving fuzzy systems from data streams in real-time", Evolving Fuzzy Systems, 2006 International Symposium on, pp. 29 – 35, 2006.

[40] D. Dovžan, V. Logar, and I. Škrjanc, "Implementation of an Evolving Fuzzy Model (eFuMo) in a Monitoring System for a Waste-Water Treatment Processe", IEEE Trans. on Fuzzy Systems, vol. 23, no. 5, pp. 1761 – 1767, 2015.

[41] D. Dovžan and I. Škrjanc, "Recursive fuzzy c-means clustering for recursive fuzzy identification of time-varying processes", ISA Transactions, vol. 50, no. 2, pp. 159 – 169, 2011.

[42] D. Lee and W. Chung, "Discrete-Status-Based Localization for Indoor Service Robots", IEEE Transactions on Industrial Electronics, Vol. 53, No. 5, pp. 1737–1746, 2006.

[43] D. Gustafson and W. Kessel, "Fuzzy clustering with fuzzy covariance matrix", in: Proc. IEEE CDC, San Diego, CA, USA, 1979, pp. 761–766.