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.2019.2931874, IEEE Transactions on Fuzzy Systems

Cluster-volume-based merging approach for incrementally evolving fuzzy Gaussian clustering - eGAUSS+

Igor Škrjanc
University of Ljubljana, Faculty of Electrical Engineering, Slovenia, igor.skrjanc@fe.uni-lj.si

***Abstract*—In this paper, a new dynamic merging approach for incrementally evolving clustering is presented. This means that the cluster partitions are incrementally learned on-line from streams of data. The criterion of merging is based on the comparison between the sum of volumes of two clusters that fulfill the criteria of a minimal number of samples in the cluster and the expected volume of the newly generated merged cluster. The newly generated merged cluster is conducted by using the weighted averaging of cluster centers and the calculation of the joint covariance matrix from the covariance matrices of the clusters. It has been shown that the proposed new evolving algorithm eGAUSS+ together with the new merging concept is very easy to implement, can work on higher-dimensional data sets, can perform all necessary computation on-line, and can produce reliable clusters.**

Keywords*:* Data Stream, Evolving Clustering, Evolving cluster models, Incremental learning, Dynamic merging, Volume of hyper-ellipsoids

I. Introduction

Mining of data streams in the context of supervised and unsupervised learning techniques has recently become an emerging research area. The methods from that area should be able to process the data step-by-step, on-line in real-time, update the parameters, and evolve the structure of the identified model. These methods are especially suitable for the processing of data that are continuously generated, are very large, and of very high dimensions. These algorithms are so-called evolving clustering algorithms, [1], [2], which are sometimes also called incremental [3], [4], [5], single-pass clustering methods [6], which means that the algorithm is working by processing the data in a step-wise way, and updating as well as evolving the structure and the parameters [7], [8]. Many of these algorithms originate from classical fuzzy clustering algorithms, such as the evolving Gustafson-Kessel algorithm in [9], [10], and [11]. The possibilistic approach to clustering proposed in [12], [13], [14], and [15] is also a great inspiration for the further development of evolving approaches based on density criteria, as proposed in [16], [17], and in [18], in which the Cauchy data distribution is assumed and [19], in which it is shown how different inner matrix norms can be used to deal with different clustering problems. The evolving principle based on principal component analysis is presented in [20]. A generalized evolving fuzzy system in an incremental single-pass manner, in which clusters appear that are of different size and rotation, is presented in [21]. A survey about evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification is given in [22]. When dealing with evolving clustering, the problem of clusters that are close and very similar frequently arises [23]. This is because the data arrive step-wise over time, and the size of the cluster is not known in advance. The two clusters can, therefore, move together and can be presented as only one. In such situations, clusters should be merged to obtain the minimum number of cluster partitions. This effect is called ”cluster fusion” and is usually caused by samples successively filling the gaps between two or more clusters, which seem to be disjointed at a former point of time in the data stream, but later it turns out that they are not, and thus should be merged to eliminate overlapping and redundant information. The merging of clusters not only provides a more accurate representation of the local data distributions but also keeps evolving neuro-fuzzy systems more compact and thus more easy to adapt and interpret.

Several different merging approaches are given in the literature. The first approaches are connected to the basic fuzzy clustering and the number of clusters. Starting with an overestimation of the number of clusters in the data, similar clusters are merged to obtain suitable partitioning. An adaptive threshold for merging is proposed in [24], [25]. This merging algorithm is guided by two geometric compatibility criteria: the first condition states that the clusters should be close enough, and the second one defines the compatibility of cluster orientation. The algorithm requires a great deal of computation and is, therefore, not suitable for the streaming data of higher dimensions.

Another approach that is able to tackle the integration of a merging concept into a k-means algorithm is given in [26], in which merged partitions are compared with original partitions based on the extended Xie-beni similarity index given in [27]; however, for each complete data set, only one cluster is merged; this means that the number of clusters is decreased by one, with the application of the merging procedure.

The proposed merging algorithms in the incremental Gaussian Mixture Models method (GMM) in [28] and [29] can solve the merging problems in chunk mode; however, the method is too slow for fast data streams in on-line learning because of a grouping of Gaussians using Chernoff bound method. The method is accelerated with a fidelity test of Gaussian using a Kolmogorov-Smirnov test in [30], but it results in an increased complexity because each new sample

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.2019.2931874, IEEE Transactions on Fuzzy Systems

creates a new Gaussian function.

The algorithm of Flexible Fuzzy Inference Systems (FLEXFIS+), given in [31], calculates the intersection of the membership functions in each dimension. This is the basis of defining the index of overlapping, which is then used to judge whether whole clusters and, consequently, the rules, should be merged or not. If the index is greater than a predefined threshold, then the clusters are merged. Merging itself is conducted in the antecedents by an extended variant of a recursive variance formula and, consequently, by exploiting Yager's participatory learning concept [32] to resolve possibly conflicting rules properly. The calculation of intersections in all dimensions is computationally demanding and not transparent, especially when dealing with higher dimensions.

In Generalized Smart Evolving Fuzzy Systems (GS-EFS), introduced in [33], two merging criteria, the touching and homogeneity condition, are used to decide whether two clusters should be merged. This means that the angles between the hyper-planes that define the local models should not be too small, and their joint volume should not expand too much. The calculation of touching criteria in all dimension is time-consuming and requires more predefined thresholds.

The evolving Fuzzy Model (eFuMo) algorithm, given in [10], merges clusters based on the normalized distance between their centers. The distance is calculated based on the Mahalanobis measure. The parameters of the merged cluster are initialized by a weighted average [34] or using a normal average, such as in [35], while the merged covariance matrix can be defined as proposed in [36]. The three conditions for merging in the eFuMo algorithm are: the angle condition, the correlation condition, and the distance ratio condition. Two clusters are merged if they fulfill one of these conditions. The algorithm becomes nontransparent for higher-dimensional problems, slow, and also the number of threshold parameters is increased.

The described merging approaches from the literature are based on geometric criteria that indicate the distances of cluster centers, the orientation of two close clusters, and overlapping. Some of them are quite complicated [10], [24], [25], very computationally intensive [10], [24], [25], [28], [29], [31], [33], can merge only two clusters at once [26], or have several conditions with predefined thresholds to be fulfilled for merging [10], [35].

The cluster merging proposed in this paper has overcome all the described weaknesses of the merging approaches known from the literature. It is based on the volume of clusters, which are calculated from the cluster covariance matrices. The method is proposed for an incremental on-line clustering and is also suitable for fast learning demands, because after each new sample only the clusters that are activated with membership degrees higher than required are checked for whether they should be merged or not. The proposed merging approach is designed to be computed in parallel. The method is simple, transparent, and easy to understand, with only one threshold parameter that has a very strong meaning, suitable for higher dimensions; it can merge more than two clusters at once, is computationally undemanding and, therefore, is very fast.

This paper is organized as follows: in the beginning, the reasons and requirements for merging techniques in evolving neuro-fuzzy systems are presented; in Section II, the Gaussian probability density distribution for modeling from data streams with recursive computation of cluster parameters, evolving Gaussian clustering, data partitioning, adding new clusters, adapting clusters with new sample, and merging the clusters are given. In Section III, some examples of that proposed method are given for clustering problems and for regression. After these examples, in Section IV, some classical benchmark examples and comparisons with some batch clustering methods are presented and discussed. At the end, the conclusion is given.

II. Gaussian probability density distribution for clustering from data streams

In our approach, the modeling from data streams is based on Gaussian probability distribution.

A. Gaussian probability density distribution

The univariate Gaussian probability density, is in general, defined as:

$$f_u(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \ , \tag{1}$$

where μ and σ^2 stand for the mean value and the variance of the data set of variable x. For the multivariate Gaussian probability density with m measured variables, the data sample is written as $z = [z_1\ z_2 \cdots z_m]^T$, the mean values of data samples inside the same cluster are written as $\mu = [\mu_1\ \mu_2 \cdots \mu_m]^T$ and the covariance matrix of the corresponding data set as Σ. The Gaussian probability density of the measured sample at the time instant k, denoted as $z(k)$, is defined as follows:

$$f_m(z(k),\mu,\Sigma) = \frac{1}{|2\pi\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(z(k)-\mu)^T\Sigma^{-1}(z(k)-\mu)} \ . \tag{2}$$

The unnormalized multivariate Gaussian density of the current sample $z(k)$ is further called "typicality" and is defined as follows

$$\gamma(k) = e^{-d^2(k)}$$

where

$$d^2(k) = \frac{1}{2}(z(k)-\mu)^T\Sigma^{-1}(z(k)-\mu) \ , \tag{3}$$

stands for the Mahalanobis distance, where

$$\mu = \frac{1}{n}\sum_{i=1}^{n} z(i) \ , \tag{4}$$

defines the center of the cluster, $z(i)$ defines the ith sample in the cluster, and n stands for the number of samples in the cluster. The Σ denotes the covariance matrix of the observed cluster, which is, in the matrix form, defined as follows:

$$\Sigma = \frac{1}{n-1}(Z-EM)^T(Z-EM) \ , \tag{5}$$

where Z stands for the data matrix of dimension $n \times m$, $Z^T = [z(1),...,z(n)]$, M stands for the matrix $M = diag(\mu_1,...,\mu_m)$ and E stands for the matrix of dimension $n \times m$ with all elements equal to 1.

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.2019.2931874, IEEE Transactions on Fuzzy Systems

The covariance matrix can be written in its singular value decomposed form as

$$\Sigma = P\Lambda P^T, \tag{6}$$

and, therefore, the Mahalanobis distance from Eq. 3 becomes as follows:

$$d^2(k) = \sum_{i=1}^{m} \frac{1}{\lambda_i}(z(k)-\mu)^T p_i \cdot p_i^T (z(k)-\mu) \tag{7}$$

where p_i stands for the ith eigenvector and λ_i for the ith eigenvalue of the covariance matrix Σ, which defines the shape of the cluster. This shows that the Mahalanobis distance is the sum of normalized distances of the current data sample from the center of the cluster in the direction of the eigenvectors. The distances are normalized with corresponding eigenvalues. The benefit of using the Mahalanobis distance is to describe the hyper-elliptically shaped cluster. The size and the shape of the hyper-ellipsoid depend on the covariance matrix of the cluster [37]. The volume of the hyper-ellipsoid in m dimensional hyper-space is defined as follows:

$$V = \frac{2\pi^{m/2}}{m\Gamma(m/2)} \Pi_{i=1}^{m} \lambda_i, \tag{8}$$

where Γ stands for the gamma function. This means that the volume of the clusters in a certain hyper-dimensional domain depends on the product of the eigenvalues of the cluster covariance matrix.

The jth cluster in the form of hyper-ellipsoids is fully represented with the triplet (μ_j, Σ_j, n_j).

B. Recursive computation of cluster parameters

The concept of evolving algorithms requires the calculation of typicalities on-line in a recursive manner. In the proposed approach, the observed measured sample is completely assigned to the cluster with the maximal typicality, if this typicality overcomes the predefined minimal value. This means that the number of samples in the cluster is incremented, and the cluster mean and the covariance matrix are adapted. Therefore, in our notation, the mean of the jth cluster with n_j samples is denoted as $\mu_j^{n_j}$, and $\Sigma_j^{n_j}$ denotes the corresponding covariance matrix.

When a new sample $z(k)$ is available, the mean and the covariance matrix of the cluster with maximal typicality are updated recursively. The update is done in the following steps. First, the difference between the current sample and the current mean value is calculated:

$$e_j(k) = z(k) - \mu_j^{n_j}. \tag{9}$$

Next, the mean is updated

$$\mu_j^{n_j+1} = \mu_j{}^{n^j} + \frac{1}{n_j+1} e_j(k). \tag{10}$$

After that, the states of the unnormalized covariance matrix are computed as:

$$S_j^{n_j+1} = S_j^{n_j} + e_j(k)\left(z(k) - \mu_j^{n_j+1}\right)^T \tag{11}$$

and the covariance matrix is then obtained as:

$$\Sigma_j^{n_j+1} = \frac{1}{n_j} S_j^{n_j+1}\ . \tag{12}$$

C. Evolving Gaussian Clustering

The evolving strategies strongly depend on the nature of

1) Data clustering: When a new incoming sample $z(k)$ is available, the typicalities to all existing clusters ($i = 1, ..., c$, c is the number of current micro-clusters and clusters together) are calculated based on the Euclidean distance,

$$\gamma_i(k) = e^{-d_i^2(k)}\ , \tag{13}$$

with

$$d_i^2(k) = (z(k)-\mu_i)^T (z(k)-\mu_i)\ , \tag{14}$$

if the number of samples in the set n_i is less than N_{max}, and based on the Mahalanobis distance

$$d_i^2(k) = (z(k)-\mu_i)^T \Sigma^{-1} (z(k)-\mu_i)\ , \tag{15}$$

if the number of samples in the set n_i is greater than N_{max}. This is to overcome the singularity problem that may appear when calculating the inverse of the covariance matrix of the cluster with too few data samples.

The Mahalanobis distance is used to detect the hyper-ellipsoidal forms and has the ability to rotate them. This means that the clusters become much more flexible and have much higher approximative ability.

The sample is assigned to the cluster (micro-cluster or cluster) with the highest typicality, denoted as $\gamma_j(k)$, if $\gamma_j(k) > \Gamma_{max}$, where Γ_{max} stands for the threshold of maximal typicality. The threshold is defined as follows

$$\Gamma_{max} = e^{-d_{max}^2(k)} \tag{16}$$

where d_{max} stands for maximal hyper-radius, which defines the resolution of the problem and should be defined according to the observed data in the following way

$$d_{max} = \min_i \left(\frac{\max z_i - \min z_i}{2N_r} \right),\ i = 1, ..., m\ , \tag{17}$$

where N_r stands for the quantization number, which is one of the tuning parameters.

2) Adding new clusters: When a new sample is available, the typicalities of this sample to all existing clusters are calculated, and then the cluster with the maximal typicality of the sample is denoted as cluster j. If this maximal typicality is lower than the defined maximal threshold, then a new cluster is formed and initialized with this sample. This means

$$\gamma_j(k) < \Gamma_{max} \tag{18}$$

where Γ_{max} stands for the user-defined maximal typicality threshold. By adding a new cluster, the current number of clusters c is incremented, the number of elements in the cluster is initialized to $n_c = 1$, and the center and the covariance matrix of the cluster are initialized to $\mu_c = z(k)$ and $\Sigma_c = \mathbf{0}$, respectively.

3) Adapting clusters with a new sample: If the maximal typicality of the current sample to the existing clusters is greater than the defined threshold,

$$\gamma_j(k) > \Gamma_{max}\ , \tag{19}$$

the sample is assigned to the cluster with maximal typicality, and the mean and the covariance matrix are adapted, as given in Eqs. 9, 10, 11, 12.

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.2019.2931874, IEEE Transactions on Fuzzy Systems

4) Merging clusters: In some cases, especially when the samples come to the learning algorithm randomly from different classes and are very dispersed, the algorithm usually creates more clusters than needed. The proposed basic evolving concept assumes a constant cluster volume measure to form new clusters, and this results in clusters of similar volume. Therefore, instead of adaptive cluster volume measure, an efficient merging approach that merges similar clusters that are close together is proposed. In this way, it decreases the number of clusters and simplifies the model structure. This means that the obtained structure can have very different clusters with very different volumes after merging.

For every cluster pair (i, j) with a triplet (μ_i, Σ_i, n_i) and (μ_j, Σ_j, n_j), the estimated joined triplet $(\mu_{ij}, \Sigma_{ij}, n_{ij})$ is calculated. The number of data samples in the joined cluster and the joined cluster center are respectively equal to

$$n_{ij} = n_i + n_j \ , \tag{20}$$

and

$$\mu_{ij} = \frac{n_i \mu_i + n_j \mu_j}{n_{ij}} \ . \tag{21}$$

The matrix of data samples for the joint cluster is now $Z_{ij}^T = \left[Z_i^T \ Z_j^T\right]$. The calculation of the covariance matrix for the joint cluster is then written as follows

$$\Sigma_{ij} = \frac{1}{n_{ij} - 1} \left(Z_i^T Z_i + Z_j^T Z_j - M_{ij}^T E_{ij}^T E_{ij} M_{ij} \right) \ . \tag{22}$$

where

$$Z_i^T Z_i = (n_i - 1)\Sigma_i + M_i^T E_i^T E_i M_i \ , \tag{23}$$

and

$$Z_j^T Z_j = (n_j - 1)\Sigma_j + M_j^T E_j^T E_j M_j \ . \tag{24}$$

This means that the covariance matrix of the joint cluster is calculated using Eqs. 20, 21, 22, 23, and 24. The proposed calculation enables the exact calculation of the joint cluster covariance matrix without storing the old data sample. The only information needed is the triples of cluster parameters.

The compactness and similarity or overlapping of two clusters (i, j) can be measured by calculating the ratio between the joined volume and the sum of both cluster volumes as follows

$$\kappa_{ij} = \frac{V_{ij}}{V_i + V_j} \ . \tag{25}$$

The ratio is called "the overlapping" and is calculated for all possible pairs of clusters that are activated with a typicality that is higher than the required activation threshold Γ_{ACT}. The activation typicality is usually defined as $\Gamma_{ACT} = \frac{1}{4}\Gamma_{MAX}$. The set of all indexes of clusters that are activated are denoted as $\mathcal{A}$ and defined as follows:

$$\mathcal{A} = \{j : \ \gamma_j > \Gamma_{ACT}, \ j = 1, ..., c\} \tag{26}$$

The most overlapped clusters are those with the minimal overlapping ratio, which should be found as follows

$$\kappa_{i^*j^*} = \min_{i,j} \kappa_{ij}, \ i, j \ \in \mathcal{A}, \tag{27}$$

and the corresponding clusters are

$$(i^*, j^*) = arg \min_{i,j} \kappa_{ij} \ , \tag{28}$$

If the minimal ratio is less than the predefined joint threshold value κ_{join}, the clusters are merged. The number of clusters is therefore decreased, $c \leftarrow c - 1$, and the corresponding clusters triplets are updated.

The complete merging algorithm is shown in Alg. 1.

Algorithm 1 Merging clusters.

```
Choice of κ_join
Initialization: merge = 1
while merge == 1
    Computation of V_i = 2π^{m/2} / (mΓ(m/2)) Π_{j=1}^m λ_j^i, i ∈ A,
    where λ_j^i stands for the jth eigenvalue of Σ_i
    For every i, j ∈ A, i ≠ j, compute
    Σ_ij = 1/(n_ij − 1) (Z_i^T Z_i + Z_j^T Z_j − M_ij^T E_ij^T E_ij M_ij) ,
    the volume V_ij, and overlapping κ_ij = V_ij / (V_i + V_j) .
    Find minimal ratio κ_{i*j*} = arg min_{i,j} κ_ij,
    if κ_{i*j*} < κ_join
        Merge clusters i* and j* in (Σ_new, μ_new, n_new)
        c ← c − 1
        merge = 1
    else
        merge = 0
    end
end
```

The complete evolving Gaussian clustering algorithm is shown in Alg. 2.

D. Estimation of the local model parameters

In the case of regression problems, the nonlinear mapping that maps a compact set from the input space of arbitrary dimension to the output space of the first dimension can be described by a number of approximators. Quite often, a Takagi-Sugeno model is used. The approaches in which the structure of the model is constant and the parameters of the model θ_j are adapted on-line are very well known.

Simultaneous model structure and parameter identification have also received a great deal of attention in the literature recently. One possible approach is to cluster the input-output data space and then identify the model parameters θ_j from the clusters, i.e. by the singular value decomposition of the covariance matrices of the clusters. This approach will be given in more detail next.

The measured data vectors are composed of the input vector u, which could be of an arbitrary dimension, and the corresponding output y. All the information about the data lies in the covariance matrices of the corresponding clusters formed in the input-output data space. It is assumed that the input-output data lie along the hyper-surface representing the input-output mapping. This assumption is usually true for a large class of problems when dealing with regression problems. Due to the nature of the processes, disturbances, measurement noise, parasitic disturbances and other sources of errors, the data do not lie exactly on the surface but are

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.2019.2931874, IEEE Transactions on Fuzzy Systems

Algorithm 2 Algorithm of Evolving Gaussian Clustering.

Choice of Γ_{max} or N_r, N_{max}
Initialization:
$c \leftarrow 1$, $\mu_c \leftarrow z(1)$, $k \leftarrow 1$
repeat $k \leftarrow k+1$
  for i = 1 : c
    if $n_i < N_{max}$
      $d_i^2(k)$ is the Euclidean distance
    else
      $d_i^2(k)$ is Mahalanobis distance
    end
    Calculation of typicality $\gamma_i(k) = e^{-d_i^2(k)}$
  end
  Choice of maximal typicality $j = arg\max_i \gamma_i(k)$
  if $\gamma_j(k) \geq \Gamma_{max}$
    Update of cluster j with new sample
    $e_j(k) \leftarrow z(k) - \mu_j$
    $\mu_j \leftarrow \mu_j + \frac{1}{n_j+1} e_j$
    $S_j = S_j + e_j(k)(z(k) - \mu_j)^T$
    $n_j \leftarrow n_j + 1$
  else
    Add and initialize new cluster
    $c \leftarrow c + 1$
    $n_c \leftarrow 1$
    $\mu_c \leftarrow z(k)$
  end
  Merging clusters
until $k > N$

spread in the vicinity of the hyper-surface. By analyzing the covariance matrices of the clusters, the models are obtained in an explicit form.

The idea originates from the definition of the hyper-plane equation in an implicit form with the normal vector of the hyper-plane and the point lying on the hyper-plane. The normal vector n_j to the hyper-surface is orthogonal to the tangential hyper-plane in the center of the cluster μ_j. This tangential hyper-plane represents the local linear model and can be obtained in the implicit equation as follows

$$(z - \mu_j)^T n_j = 0 \tag{29}$$

The normal vector is defined as

$$n_j = p_j^r \tag{30}$$

where p_j^r denotes the first eigenvector of the covariance matrix Σ^j that has an eigenvalue close to zero (close to noise variance). For example, if there is one regressor in the regressor matrix that is linearly dependent on another regressor, the qth eigenvalue should be ≈ 0 and the $(q-1)$th eigenvalue should reflect the noise. Therefore, in this case, $r = q - 1$.

In the case of regression problems, the regressors are usually very carefully chosen, meaning that they are linearly independent, and the excitation of the process is adequate. In this way, the rank of the current covariance matrix is $q - 1$, which means that only one eigenvalue of the matrix is close to zero (close to noise variance). In this case, the normal vector n_j is equal to the latent eigenvector p_j^q.

Because of the nonlinear nature of the data, the normal vector to the hyper-surface changes from one operating point to another. In the context of fuzzy approximators, the normal vector in a certain operating point is obtained by a linear combination of individual normal vectors associated with individual clusters. The gains of the linear combination are obtained from the typicality or membership degree of the individual clusters. Here, the typicalities associated with the clusters are used. This leads to the following estimation of the model output:

$$y(k) = \frac{\sum_{j=1}^{c} \gamma_j(k) y_j(k)}{\sum_{j=1}^{c} \gamma_j(k)} \tag{31}$$

where c stands for the number of clusters. The problem here arises because the typicality $\gamma_j(k)$ depends on the data sample $z(k)$, which is not known in the case of the usual use of the model. More precisely, the first part of z (the input vector u) is known, but the output y is unknown. This problem can be solved by projecting the typicality function to the input space or by replacing the output value by the jth local model output as follows:

$$\bar{z}_j^T(k) = \begin{bmatrix} u^T(k) & \bar{y}_j(k) \end{bmatrix} \tag{32}$$

The typicality can be then calculated as follows

$$\gamma_j(k) = e^{-\frac{1}{2}(\bar{z}_j(k) - \mu_j)^T (\Sigma_j)^{-1} (\bar{z}_j(k) - \mu_j)} \tag{33}$$

and this projected membership function is called $\mathcal{Z}_j$.

III. Examples

In this section, the results of the algorithm are shown on simple examples of clustering and regression problems.

A. Clustering

The test of the evolving Gaussian clustering algorithm was first done using the artificial data of two dimensions, in which the data come randomly from three different stochastic processes. The first process generates the following data $z_{11}(k) = \mathcal{N}(0,1)$ and $z_{12}(k) = z_{11}(k) + \mathcal{N}(0, 0.8)$, with 250 data samples from this process. With $\mathcal{N}(\mu, \sigma)$, the Gaussian normal noise is described, where μ stands for the mean value, and σ for the standard deviation of white noise. The second process generates the data with the following values $z_{21}(k) = \mathcal{N}(0,1)$ and $z_{22}(k) = -z_{21}(k) + \mathcal{N}(8, 0.85)$, with 100 data samples, and the third process generates the data $z_{31}(k) = \mathcal{N}(-1.5, 0.5)$ and $z_{32}(k) = -z_{32}(k) + \mathcal{N}(4, 0.5)$, with 50 data samples. The data stream is shown in Fig. 1, where it is evident that the data are coming randomly. The algorithm was initialized with the following tuning parameters: the parameter $\kappa_{join} = 1.0$, the maximal number to use the Euclidean distance measure is $N_{max} = 4$, the initial grid is defined by $N_r = 4$; this means that $\Gamma_{max} = 0.66$, and the clusters with less than two samples are removed. The data samples in the final situation and final clusters with centers and 3σ contour are shown in Fig. 2. The contours are drawn for at least 4 samples in a cluster. Changing the merging parameter

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.2019.2931874, IEEE Transactions on Fuzzy Systems

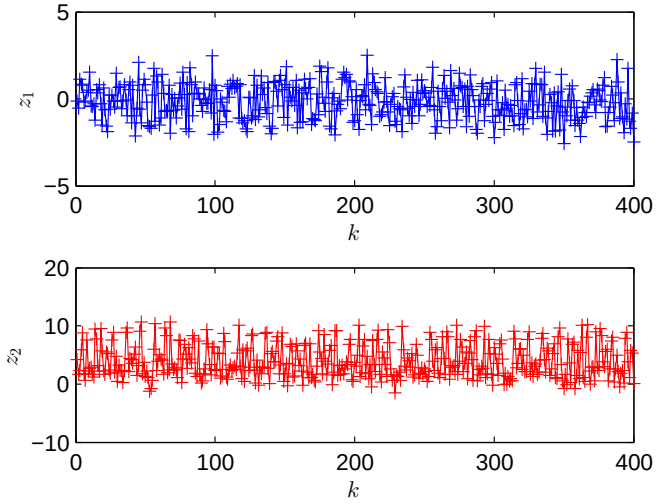

Fig. 1. Random data stream from three different stochastic processes.

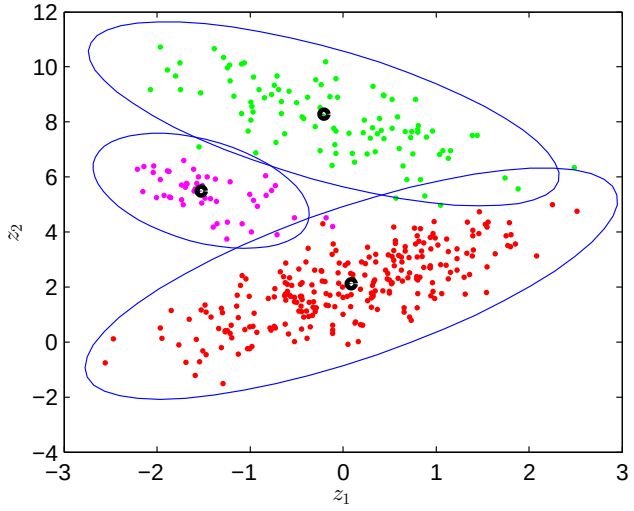

Fig. 3. The data and final clusters with centers and 3σ contour, and $\kappa_{join} = 1.4$.

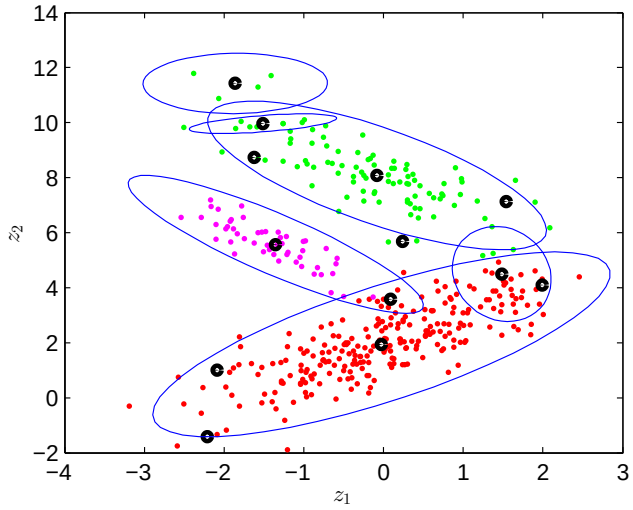

Fig. 2. The data and final clusters with centers, 3σ contour, and $\kappa_{join} = 1.0$.

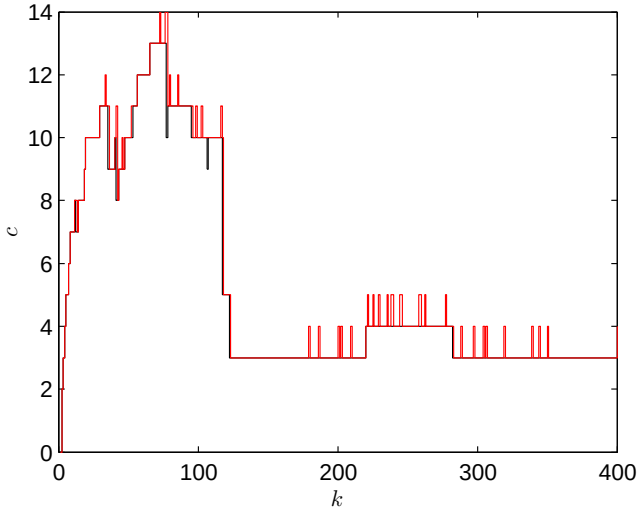

Fig. 4. The time course of the number of clusters.

to $\kappa_{join} = 1.4$ means much less restrictive merging criteria and results in just three clusters, as shown in Fig. 3. In Fig. 4, the time course of the cluster number is presented, in which red indicates the number before merging and black the number of clusters after the merging procedure. It is shown that the maximal number of detected clusters was 14, and it dropped to 3 at the end.

B. Clustering for switching process data

In the second test, a switching linear process was studied. The data stream is formed as follows: first, the independent variable $z_1(k) = \mathcal{N}(0,1)$ of $n = 200$ samples is generated. Then the switch variable $s(k) = \mathcal{N}(0,1)$ is generated. The value of the dependent variable depends on the switching value. If $s(k) > 0$ then $z_2(k) = z_1(k) + \mathcal{N}(0, 0.05)$, else $z_2(k) = \mathcal{N}(0, 0.05)$. The data stream is shown in Fig. 5, where it can be clearly seen that the data are randomly coming from these two processes. The algorithm was initialized with the following tuning parameters: the parameter $\kappa_{join} = 1.1$, the maximal number to use the Euclidean distance measure equals $N_{max} = 4$, and the initial grid is defined by $N_r = 12$. The data samples in the final situation and final clusters with centers and 3σ contour are shown in Fig. 6. Next, the change of merging parameter to $\kappa_{join} = 1.4$ is realized. In that case, the clusters are merged even when the overlapping is smaller; therefore, the number of clusters is much lower, as shown in Fig. 7. In Fig. 8, the time course of the number of clusters is presented, where red indicates the number before merging and black the number of clusters after the merging procedure. It is shown that the clustering at the beginning is rather fine and, at the end, only the natural clusters are obtained.

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.2019.2931874, IEEE Transactions on Fuzzy Systems

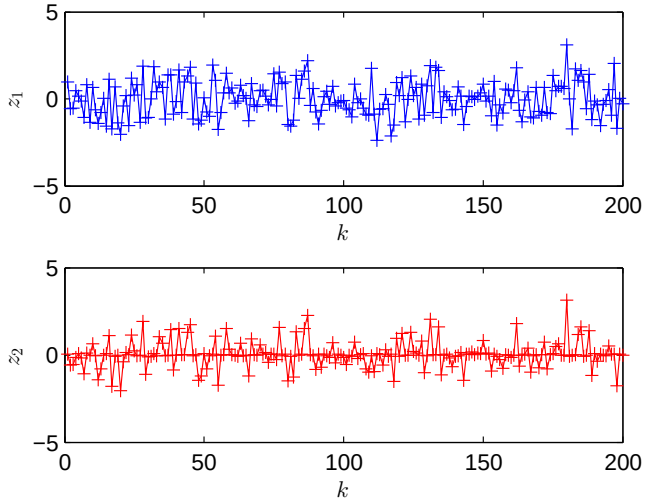

Fig. 5. Random data stream from three different stochastic processes.

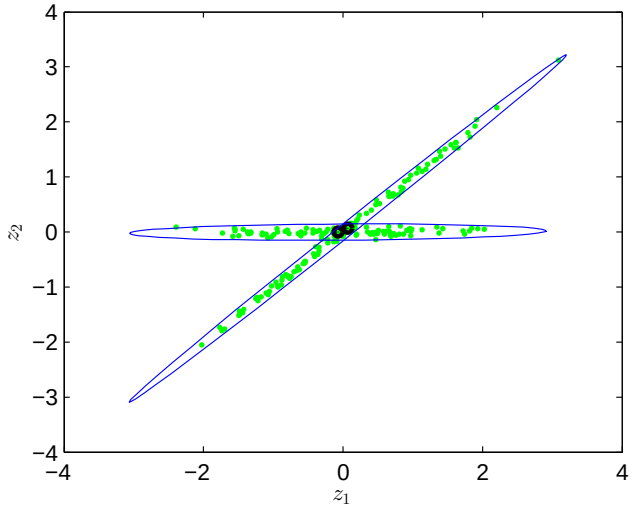

Fig. 7. The data and final clusters with centers and 3σ contour, and $\kappa_{join} = 1.4$.

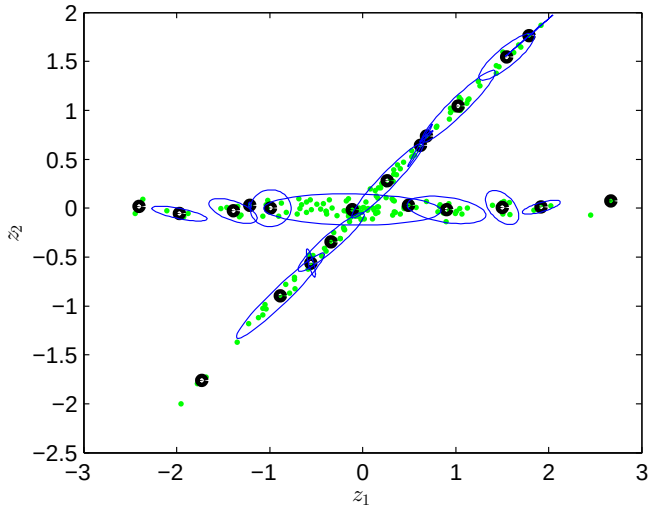

Fig. 6. The data and final clusters with centers, 3σ contour, and $\kappa_{join} = 1.1$.

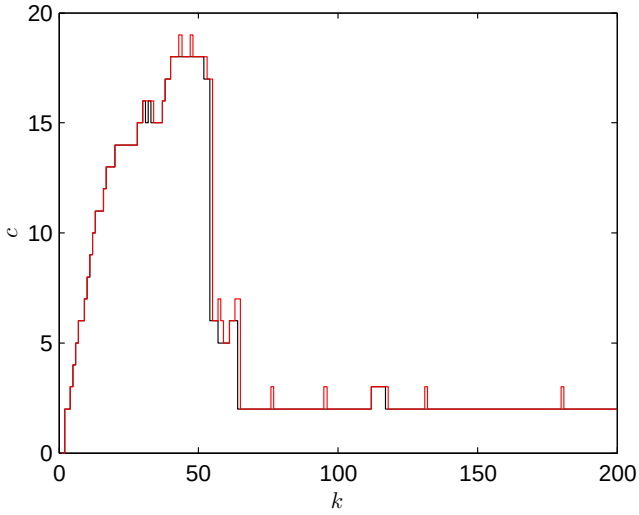

Fig. 8. The number of clusters.

C. Clustering for regression

In the example of using the proposed algorithm in regression problems, the data stream is formed as follows: first, the independent variable $z_1(k) = \mathcal{N}(0,1)$ of $n = 400$ samples is generated. The value of the dependent variable is a nonlinear function, $z_2(k) = 0.4z_1^3(k) + \mathcal{N}(0, 0.2)$.

The algorithm is initialized with the following tuning parameters: the parameter $\kappa_{join} = 1.25$, the maximal number to use the Euclidean distance measure is $N_{max} = 5$, the initial grid is defined by $N_r = 4$, and the minimal number of samples in a cluster to remove it is defined as $N_{min} = 4$. The data samples in the final situation and final clusters with centers and 3σ contour are shown in Fig. 9. The removed clusters are shown only with centers of clusters.

Changing the merging parameter to $\kappa_{join} = 0.95$, the clusters are merged with more volume overlapping. In Fig. 10, the result with more clusters is shown.

The merging parameter $\kappa_{join} = 1.8$ results in merging with less volume overlapping, therefore with fewer final clusters, which is shown in Fig. 11.

The clusters are defined by the center μ_j and the covariance matrix Σ_j. In regression problems, local linear models must also be identified, either by the recursive least-squares method or by defining the hyper-plane that spans the data. The results of this approach are shown next.

The approximation model in Takagi-Sugeno form, for example, from Fig. 11, is now described as follows

$$R_1: \ \textbf{if } z_1 \textbf{ is } \mathcal{Z}_1 \textbf{ then } z_2 = 4.47z_1 - 10.33 \quad (34)$$

$$R_2: \ \textbf{if } z_1 \textbf{ is } \mathcal{Z}_2 \textbf{ then } z_2 = 0.41z_1 + 0.05 \quad (35)$$

$$R_3: \ \textbf{if } z_1 \textbf{ is } \mathcal{Z}_3 \textbf{ then } z_2 = 3.76z_1 - 8.36 \quad (36)$$

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.2019.2931874, IEEE Transactions on Fuzzy Systems

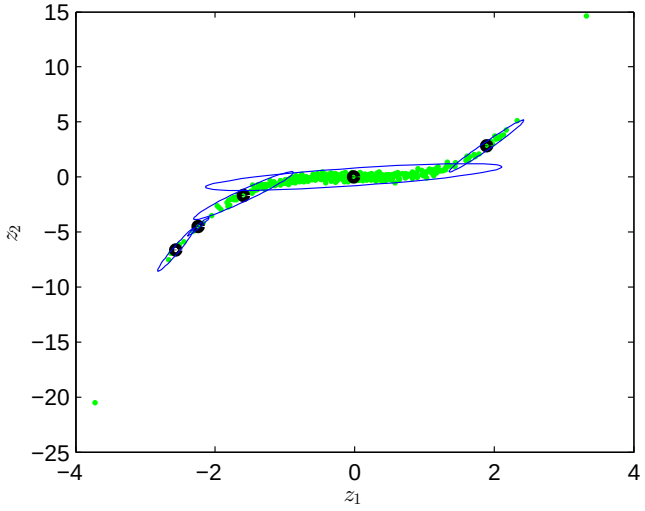

Fig. 9. The data and final clusters with centers and 3σ contour, and $\kappa_{join} = 1.25$.

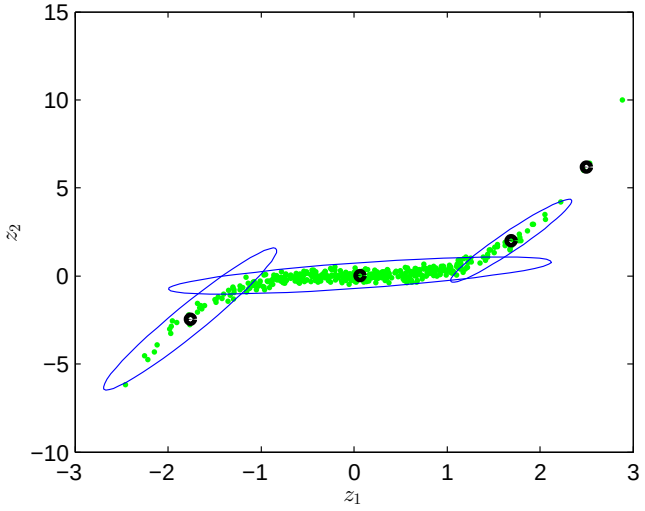

Fig. 11. The data and final clusters with centers, 3σ contour, and $\kappa_{join} = 1.8$.

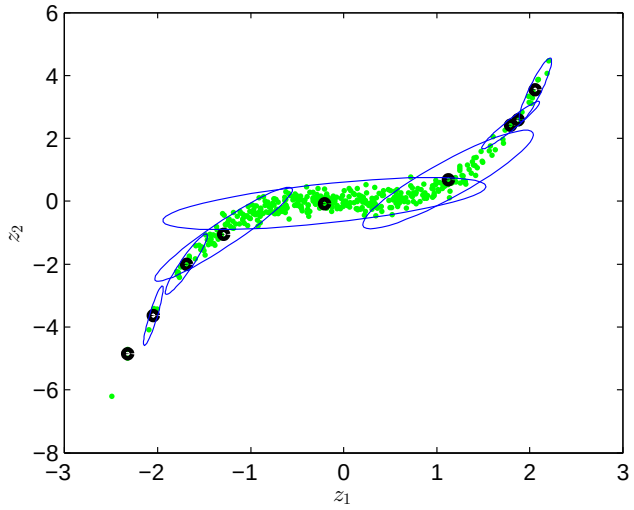

Fig. 10. The data and final clusters with centers, 3σ contour and, $\kappa_{join} = 0.95$.

IV. Benchmark examples and comparisons

In general, the proposed method is compared to multi-pass clustering methods, such as LOLIMOT (LOLI) [38], HILOMOT [39] (HILO), SUHICLUST [40], [41] (SUHI) and FUZZYID [42]. The comparison to multi-pass methods and not to single-pass methods is done because the former usually yields better results. The methods were compared according to the number of local models (#LMs), the normalized root mean square error in the case of learning data set (NRMSE_L), the normalized root mean square error in the case of testing data set (NRMSE_T), and the computation time in seconds (t[s]). The normalized root mean square error (NRMSE) is defined as follows:

$$NRMSE = \left(\frac{\frac{1}{N}\sum_{k=1}^{N}(y(k) - y_m(k))^2}{\frac{1}{N}\sum_{k=1}^{N}(y(k) - \overline{y})^2} \right)^{\frac{1}{2}}, j = 1, ..., r,$$
$$\overline{y} = \frac{1}{N}\sum_{k=1}^{N} y(k) \tag{37}$$

which means that the sum of square errors between the measured variable $y(k)$ and the modelled variable $y_m(k)$ is normalized according to the variance of the measure variable. The methods were tested on a laptop with an i7, 2.9 GHz processor and 16GB RAM.

All the results are presented in the form of tables, in which the first column of the table represents the method's name. In the second column follows the number of generated local models. The third column represents the error on the learning set and the fourth the error on the testing set. In the fourth column, the computation time is given.

A. Static examples

Three examples are chosen for the comparison of static problems: the MARS 1 problem [43], the prediction of the Mackey-Glass time series [44], and the Hyperbola-benchmark [41].

1) The MARS 1 example: The MARS 1 is the function given as follows:

$$y = \frac{2e^{f_1}}{e^{f_2} + e^{f_3}} \tag{38}$$

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 is 900 samples, for which u_1 and u_2 are randomly generated from a uniform distribution. The validation data also has 900 samples; however, in this case, u_1 and u_2 are equally distributed from zero to one, creating a grid.

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.2019.2931874, IEEE Transactions on Fuzzy Systems

TABLE I
RESULTS FOR MARS PROBLEM

Method	#LMs	NRMSE$_L$	NRMSE$_T$	t[s]
eGAUSS+	**5**	**0.0327**	**0.0422**	< 1
LOLI	20	0.0495	0.0686	≈ 3
LOLI $C\#$	11	0.0452	0.0528	≈ 1
HILO	6	0.0489	0.0502	≈ 4
SUHI	9	0.0474	0.0524	≈ 4
FUZZYID	25	0.0567	0.0534	< 1

TABLE II
RESULTS FOR MG SERIES PREDICTION PROBLEM

Method	#LMs	NRMSE$_L$	NRMSE$_T$	t[s]
eGAUSS+	**24**	0.0481	0.0621	≈ 5
LOLI	31	0.1362	0.1562	≈ 6
LOLI $C\#$	31	0.0498	0.0649	≈ 20
HILO	30	0.0478	0.0618	≈ 26
SUHI	29	**0.0477**	**0.0577**	≈ 197
FUZZYID	81	0.059	0.0771	≈ 17

The goal is to achieve an NRMSE$_L$ of 0.05. The global least-squares method is used to estimate the local model parameters for all fuzzy methods. The obtained results are presented in Table I. In the case of the eGAUSS+ method, the results are obtained with the parameter $\kappa_{join} = 1.3$.

2) Mackey-Glass time series: The chaotic time series [44] is generated from the Mackey-Glass (MG) differential delay equation defined by the following equation:

$$z(t) = \frac{0.2z(t-\tau)}{1+z^{10}(t-\tau)} - 0.1z(t) \tag{39}$$

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:

$$\begin{aligned} y(k) &= [z(k+85)] \\ \boldsymbol{u}(k) &= [z(k-18)\ z(k-12)\ z(k-6)\ z(k)] \end{aligned} \tag{40}$$

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. Very good performances were obtained with eGAUSS+, LOLI $C\#$, and SUHICLUST. The best normalized root-square error (NRMSE$_T$) is, in this case, obtained with the SUHICLUST method; however, that method produced five more local models than eGAUSS+ did, and the learning time with SUHICLUST is considerably longer. The results with the eGAUSS+ method are obtained with the parameter $\kappa_{join} = 1.25$.

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

$$y = \frac{1}{0.1 + \frac{1}{p}\sum_{i=1}^{p}(1-u_i)} \tag{41}$$

The tests are made for 1-, 4-, 7-, and 10-D input space. The inputs are randomly generated from an interval $[0, 1]$. For learning, 900 samples are used, and 2000 samples for testing. The number of generated local models is presented in Table III. For the one-dimensional problem, all methods perform reasonably well. When the dimension is increased, the FUZZYID has a substantial problem with the dimensionality. The FUZZYID method failed to generate a model for a 10-D problem. Also, the control of the number of local models is impossible for this method. Since they use grid partitioning, the number of local models increased exponentially with the dimension of the data space.

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

	1-D	4-D	7-D	10-D
Method	#LMs			
eGAUSS+	4	4	3	3
LOLI	5	22	32	29
LOLI $C\#$	4	10	14	18
HILO	4	4	4	4
SUHI	5	5	5	3
FUZZYID	5	16	128	/

An interesting observation can be made by examining the number of generated local models. It can be seen that the number of local models increased with the space dimension with the LOLIMOT method, while with the eGAUSS+ ($\kappa_{join} = 1.2$) and SUHICLUST methods the number of local models decreased or at least remained 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 0.08 for the 10-D problem. The decrease of variance means that the span of output is lower; therefore, the non-linearity also decreases. This means that fewer local models are needed to approximate the model.

B. Dynamic examples

Two examples are chosen for a dynamic modeling problem: the Coupled Electrical drive example (CED) [45] and the Cascade Tanks Example (TCT) [45]. As seen from previous static examples, the FUZZYID has problems with high dimensional examples; therefore, it is omitted from the dynamic 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 goal, in this case, is to identify the function that relates the process input, which is the sum of the voltages applied to the motors, and the output, which is the pulley velocity [45]. According to [46], the regressor vector in the form of $\boldsymbol{u}(k) = [1, y(k-1), y(k-4), y(k-6), y(k-7), y(k-8), y(k-9), u(k-1), u(k-4), u(k-6)]$ is chosen. The space for partitioning is 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 the identification are presented in Table IV. In this example, the target NRMSE$_L$ error for the HILO, SUHI, and LOLI

TABLE IV
RESULTS FOR CED

Method	#LMs	NRMSE$_L$	NRMSE$_T$	t[s]
eGAUSS+	**4**	**0.1067**	**0.1326**	< 1
LOLI	12	0.1157	0.3595	≈ 3
LOLI $C\#$	7	0.1194	0.3375	≈ 1
HILO	7	0.5750	0.4250	≈ 11
SUHI	6	0.1079	0.1666	< 1

TABLE V
RESULTS FOR TCT

Method	#LMs	NRMSE$_L$	NRMSE$_T$	t[s]
eGAUSS+	**4**	**0.0428**	0.0445	< 1
LOLI	7	0.0438	**0.0406**	≈ 3
LOLI $C\#$	5	0.0463	0.0517	≈ 1
HILO	**4**	0.0498	0.0518	≈ 8
SUHI	**4**	0.0485	0.0533	≈ 8

methods is set to 0.12. The eGAUSS+ method reached the threshold with only four local models and has the lowest test NRMSE. While the HILOMOT produces 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.

2) Two cascaded tanks: This is the process of the liquid level control system. It consists of two cascaded tanks with free outlets fed by a pump. The liquid (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 deep with a large vertical extension, this results in a significantly non-linear behaviour that varies with the level of water [45]. The goal here is to identify the level h_2. According to [46], the regressor vector in the form of $\boldsymbol{u}(k) = [1, h_2(k-4), h_2(k-1), h_1(k-4), u(k-2), u(k-4)]$ is chosen. The space for partitioning is 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 NRMSE$_L$ for the HILO, SUHI, LOLI, and eGAUSS+ methods is set to 0.05. In this example, the HILOMOT produces a valid model with the same number of local models as eGAUSS+ and SUHICLUST, with eGAUSS+ having a slightly better test NRMSE$_T$ value.

V. CONCLUSION

A new merging approach based on cluster-volume for incrementally evolving Gaussian clustering is proposed for achieving more efficient data stream clustering. This dynamic cluster merging is based on the cluster volume and the cluster covariance matrix and is suitable for very fast learning demands in real time, because the algorithm verifies whether the clusters that are activated with a certain level should be merged or not. This decreases the computational demand and the computation time, which is the major problem of known merging approaches from the literature.

The results of comparisons show that the evolving Gaussian model together with the proposed merging approach is able to define the cluster partitions more reliably than well-known multi-pass fuzzy clustering methods are, and they come much closer to the natural number of clusters.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Angelov P.P., and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams", *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1462 – 1475, 2008.

[2] Angelov P.P., E. Lughofer, and X. Zhou, "Evolving Fuzzy Classifiers using Different Model Architectures", *Fuzzy Sets and Systems*, vol. 159, no. 23, pp. 3160 – 3182, 2008.

[3] Wang Y., L. Chen, and J.P. Mei,"Incremental Fuzzy Clustering With Multiple Medoids for Large Data ", *IEEE Trans., on Fuzzy Systems*, vol. 22, no. 6, pp. 1557 – 1568, 2014.

[4] Havens T.C., J. Bezdek, C. Leckie, L. Hall, and M. Palaniswami, "Fuzzy c-means algorithms for very large data", *IEEE Trans. on Fuzzy Systems*, vol. 20, no. 6, pp. 1130–1146, 2012.

[5] Lughofer, E., "FLEXFIS: A Robust Incremental Learning Approach for Evolving TakagiSugeno Fuzzy Models", *IEEE Trans. on Fuzzy Systems*, vol. 16, no. 6, pp. 1393-1410, 2008.

[6] Hall L., D.B. Goldof, "Convergence of the Single-Pass and Online Fuzzy C-Means Algorithms", IEEE Trans, of Fuzzy Systems, vol. 19, no. 4, pp. 792–794, 2011.

[7] Angelov P., "An approach for fuzzy rule-base adaptation using on-line clustering", *Int. J. Approx. Reasoning*, vol. 35, no. 3, pp. 275-289, 2004.

[8] Angelov P, D. Filev, and N. Kasabov, "Evolving intelligent systemsmethodology and applications", *Wiley*, New York, 2010.

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

[10] Dovžan D., V. Logar, and I. Škrjanc, "Implementation of an Evolving Fuzzy Model (eFuMo) in a Monitoring System for a Waste Water Treatment Process", *IEEE Trans. on Fuzzy Systems*, vol. 23, no. 5, pp. 1761–1776, 2015.

[11] Filev D., 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, 2010.

[12] Krishnapuram R., and J. M. Keller, "Possibilistic approach to clustering", *IEEE Trans. on Fuzzy Systems*, vol.1, no. 2, pp. 98 – 100, 1993.

[13] Ojeda-Magana B., R. Ruelas, M. A. Corona-Nakamura, and D. Andina, "An improvement to the possibilistic fuzzy c-means clustering algorithm", *Intelligent Automation and Soft Computing*, vol. 20, no. 1., pp. 585 – 592, 2006.

[14] Pal N.R., K. Pal, J. M. Keller, and J. C. Bezdek, "A possibilistic fuzzy c-means clustering algorithm", *IEEE Trans. on Fuzzy Systems*, pp. 517 – 530, vol.13, no. 4, 2005.

[15] Timm H., C. Borgelt, C. Doering, and R. Kruse, "An extension to possibilistic fuzzy cluster analysis", *Fuzzy Sets and Systems*, vol. 147, no. 1., pp. 3-16, 2004.

[16] Hathaway R.J., and Y. Hu, "Density-weighted fuzzy c-means clustering", *IEEE Trans. on Fuzzy Systems*, vol. 17, no. 1, pp. 243–252, 2009.

[17] Angelov P.P., and R. Yager, "Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density", *2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*, pp. 62-69, 2011.

[18] Škrjanc I., 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.

[19] Škrjanc I., 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, 2018.

[20] Klančar G. 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.
[21] Lughofer E., M. Pratama, and I. Škrjanc, "Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation", *IEEE Trans. on Fuzzy Systems*, vol. 26, no. 4, pp. 1854–1865, 2018.
[22] Škrjanc I., J.A. Iglesias, A. Sanchis, D. Leite, E. Lughofer, F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification : a survey", *Information sciences*, ISSN 0020-0255. [Print ed.], In Press, 2019.
[23] Lughofer E. and M. Sayed-Mouchaweh, "Autonomous data stream clustering implementing incremental split-and-merge techniques — towards a plug-and-play approach",*Info Sci*, vol. 204, pp. 54–79, 2015.
[24] Kaymak U., and M. Setnes, "Fuzzy Clustering With Volume Prototypes and Adaptive Cluster Merging", *IEEE Trans. on Fuzzy Systems*, vol. 10, no. 6, pp. 705–712, 2002.
[25] Kaymak U., and R. Babuska, "Compatible cluster merging for fuzzy modelling", Proceedings of 1995 IEEE International Conference on Fuzzy Systems., Yokohama, Japan, pp. 897–904 vol.2, 1995.
[26] Beringer J., E. Hüllermeier, "Online clustering of parallel data streams", *Data Knowl. Eng.*, vol. 58, no. 2, pp. 180-204, 2007.
[27] Xie X.L., G. Beni, "A validity measure for fuzzy clustering", *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 13, no. 48, pp. 841-847, 1991.
[28] Hall P, Hicks Y (2005) A method to add gaussian mixture models. *Tech. rep., University of Bath, Tech. Rep.*, 2005.
[29] Song M, H. Wang, "Highly efficient incremental estimation of gaussian mixture models for online data stream clustering",In: *Priddy KL (ed) Intelligent computing: theory and applications III. In: Proceedings of the SPIE*, vol. 5803, pp. 174-183, 2005.
[30] Declercq A. and J. Piater, "Online learning of gaussian mixture modelsa two-level approach", In: *Proceedings of the 3rd international conference on computer vision theory and applications VISAPP*, Funchal, Portugal, pp. 605-611, 2008.
[31] Lughofer E., J.-L. Bouchot, and A. Shaker, "Online elimination of local redundancies in evolving fuzzy systems", *Evolving Systems*, no. 2, pp. 165–187, 2011.
[32] Yager R. R., "A model of participatory learning",*IEEE Trans. on Systems, Man and Cybernetics*, vol. 20, no. 5, pp. 1229–1234, 1990.
[33] Lughofer E., C. Cernuda, S. Kindermann, and M. Pratama, "Generalized smart evolving fuzzy systems", *Evolving Systems*, vol. 6, no. 4, pp. 269–292, 2015.
[34] Soleimani-B H., C. Lucas, and B. N. Araabi, "Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling", *Evolving Systems*, vol. 1, no. 1, pp. 59–71, 2010.
[35] Kasabov N., "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning", *IEEE Trans. on Systems Man and Cybernetics - Part B*, vol. 31, no. 6, pp. 902–918, 2001.
[36] Li W., H. H. Yue, S. Valle-Cervantes, and S. J. Qin, "Recursive PCA for adaptive process monitoring", *Journal of Process Control*, vol. 10, no. 5, pp. 471–486, 2000.
[37] Lughofer E., "A dynamic split-and-merge approach for evolving cluster models", *Evolving Systems*, vol. 3, no. 3, pp. 135–151, 2012.
[38] Nelles O., S. Sinsel and R. Isermann, "Local basis function networks for identification of turbocharger", *IEEE UKACC Int. Conf. Control*, Exeter, pp. 7–12, UK, 1993.
[39] Hartmann B., T. Ebert, T. Fischer, J. Belz, G. Kampmann and O. Nelles, LMNtool - Toolbox zum automatischen Trainieren lokaler Modellnetze, *22th Workshop Computational Intelligence*, Dortmund, 2012.
[40] Teslić L., 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 Trans. on Neural Networks*, vol. 22, no. 12, pp. 1941–1951, 2011.
[41] Škrjanc I., "Evolving Fuzzy-Model-Based Design of Experiments With Supervised Hierarchical Clustering", *IEEE Trans. on Fuzzy Systems*, vol. 23, no. 4, pp. 861 - 871, 2014.
[42] Abonyi J., "Fuzzy Model Identification for Control", Birkhäuser, Boston, 2003.
[43] Friedman J.H., "Multivariate adaptive regression splines", *Ann. Statist.*, vol. 19, no. 1, pp. 1–141, 1991.
[44] Mackey M.C. and L. Glass, "Oscillations and chaos in physiological control systems", *Science*, vol. 9, pp. 197–287, 1977.
[45] Wigren T. and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification", *IEEE European Control Conference*, pp. 2933-2938, 2013.
[46] Aleksovski D., Dovžan D., Džeroski S., and J. Kocijan, "A comparison of fuzzy identification methods on benchmark datasets", *IFAC-PapersOnLine*, vol. 49, no. 5, pp. 31 – 36, 2016.

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 Department for Control Systems and Cybernetics. His main research interests include intelligent, predictive control systems, and autonomous mobile systems.

In 2008 he received the Highest Research Award of the Republic of Slovenia for Scientific and Research Achievements, 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, and the JSPS research fellowship for year 2017. He is also Chair of Excellence at University Carlos III in Madrid, Spain.