Information Sciences 478 (2019) 540–563

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Inner matrix norms in evolving Cauchy possibilistic clustering for classification and regression from data streams

Igor Škrjanc[a,*], Sašo Blažič[a], Edwin Lughofer[b], Dejan Dovžan[a]

[a] Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, Ljubljana, SI-1000, Slovenia
[b] Department of Knowledge-Based Mathematical Systems, Johannes Kepler University, Altenbergerstrasse 69, Linz, Austria

ARTICLE INFO

Article history:
Received 6 June 2018
Revised 11 November 2018
Accepted 17 November 2018
Available online 22 November 2018

Keywords:
Data stream
Evolving clustering
Cauchy density

ABSTRACT

This paper presents the unification and generalization of different evolving clustering methods based on Cauchy density. This can be done by introducing different inner matrix norms to obtain different functionalities of the algorithm. This unified approach with a general inner matrix norm is called eCauchy recursive clustering. The well-known possibilistic c-means clustering (PCM) can be seen as a special example of the proposed eCauchy algorithm. The main motivation of the proposed method is to solve and overcome the problems of modelling the nonlinear data streams in highly noisy environments with frequently appearing outliers. By introducing the different inner matrix into the density metric, the algorithm can be modified in different ways to deal with different clustering problems, from classical classification to the preprocessing for solving regression problems. The evolving nature of the algorithm and simple computation also make it appropriate for dealing with big-data problems. The described eCauchy algorithm needs just a few initial parameters such as minimal and maximal density. The algorithm incrementally changes the structure of the model based on the flow of samples from the data stream, more specifically it evolves the structure of the model during the operation by adding, merging, splitting and removing the clusters. This approach allows the identification of very different clusters in size and shape and is also quite insensitive to the outliers and significant noise. In the paper, the universality of the proposed algorithm is shown on various examples.

1. Introduction

The basic fuzzy c-means clustering (FCM) given in [13] and generalized in [8] cannot deal with data that include many outliers and significant noise. The main reason is the concept of membership values, which is based on normalized Euclidean distance between the current sample and the prototypes [1]. The membership is a relative measure because the distance between the sample and the prototype is normalized by the sum of the distances from all cluster prototypes. In [35], the membership value is therefore interpreted as a relative measure or relative typicality. The absolute typicality of the sample is calculated without normalization.

* Corresponding author.
E-mail addresses: igor.skrjanc@fe.uni-lj.si (I. Škrjanc), edwin.lughofer@jku.at (D. Dovžan).

https://doi.org/10.1016/j.ins.2018.11.040
0020-0255/

1.1. Possibilistic clustering

In [24], the possibilistic c-means (PCM) clustering algorithm was introduced to overcome the disadvantage of FCM and in [45] an extension of the algorithm has been made, which introduce a mutual repulsion of the clusters, so that they are forced away from each other. The main idea of this approach is to overcome the problems connected to the use of relative membership measure, by introducing a similarity measure between the observed sample and the prototypes, which is given as an absolute measure or typicality. All the samples that have typicality greater than the predefined threshold are typical samples of the observed cluster and therefore belong to the cluster, whereas the others are atypical and do not belong to the cluster. The algorithm has some disadvantages, such as sensitivity to initialization and the possible coincidence of several prototypes in the same location when the assumed number of clusters exceeds the actual number of clusters in the dataset.

In [35], a hybrid algorithm is proposed that combines the FCM and PCM approach. The algorithm is called possibilistic fuzzy c-means clustering (PFCM). This is an attempt to combine both algorithms (PCM and FCM) to estimate the prototypes of data sets as a function of the internal resemblance and external dissimilarity.

The modification, which makes the algorithm much more flexible in the sense of detecting different shapes, is given in [34] by introducing the Mahalanobis distance as an inner norm. This modification allows for detecting the hyper-ellipsoidal form of the clusters. The implementation of Mahalanobis distance as the similarity measure relativizes the absolute typicality involved in the PCM algorithm.

1.2. Evolving recursive clustering for streaming data

All previously mentioned algorithms deal with batch data. To deal with the stream of data on-line the algorithms must be able to adapt and evolve the structure of a model in an on-line/recursive manner. To accomplish that, an on-line clustering algorithm and local model parameters adaptation algorithms are needed. The recursive partitioning/clustering algorithms are usually derived from off-line clustering algorithms. In [14] and in [11], an on-line Gustafson–Kessel clustering algorithm is presented, in [2] an on-line version of subtractive clustering technique is given, in [42] a recursive method based on Gath–Geva clustering algorithm is presented, and in [33] a recursive possibilistic clustering algorithm is derived.

The on-line partitioning/clustering technique together with some sort of on-line adaptation of consequent parts form an on-line learning algorithm of the fuzzy model. Early development of the on-line fuzzy model learning techniques did not adapt the parameters of membership functions. This was introduced by evolving fuzzy systems. This new methodology was able to add new clusters to fuzzy model structure but did not adapt the membership functions' widths. The method was further extended to allow the adaptation of membership functions' widths. These extensions are known as the evolving extended Takagi–Sugeno (exTS) based neuro-fuzzy algorithm and the modification of evolving Takagi–Sugeno (eTS+) algorithm [3].

The on-line clustering algorithms are mostly based on the Euclidean distance [10]. This results in clusters that are of a hyper-spherical shape. Additional improvement was made by the introduction of Mahalanobis distance in the recursive and evolving fuzzy model identification and clustering [11,14,31]. More information on evolving approaches and on-line clustering can be found in [30].

The algorithm, which is an extension of the basic PCM, is given in [44] and is capable of dealing with different cluster shapes, sizes, volumes as well as noisy data with outliers in an on-line manner. It is called Gustafson–Kessel Possibilistic Fuzzy c-Means Clustering Algorithm (GKPFCM) because the structure of the fuzzy model is evolved with an incoming data stream. The clusters are added, merged, split and/or deleted if certain conditions are satisfied. The algorithm adapts parameters of the current structure if the current sample is typical for a particular, already identified cluster, i.e. the mean and the covariance matrix of that cluster are adapted accordingly. The on-line self-organizing fuzzy modified least-squares network (SOFMLS) is given in [20,25,41], and a sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction in [38].

Recently, the concept of granules and later clouds was introduced in [4]. This concept is based on Cauchy density, but it is not limited just to this similarity measure. The concept was further extended in [6], where the algorithm for typicality and eccentricity data analytics (TEDA) is introduced. This method introduces the typicality distribution functions as an alternative to probability density functions in [4]. An algorithm called parsimonious network based on fuzzy inference system (PANFIS) is presented in [36]. The main feature of PANFIS is that it can commence its learning process from scratch with an empty rule base. The evolving participatory learning (ePL), introduced in [26], use on-line clustering phase followed by least square method to estimate the linear consequent parameters of Takagi–Sugeno model. The generalized smart evolving fuzzy systems learning approach (Gen-Smart-EFS, GS-EFS) employs the generalized version of Takagi–Sugeno (TS) fuzzy systems [31]. It induces more compact rule bases with similar or even lower model errors than the conventional TS fuzzy systems.

In this paper the Cauchy density was generalized in a way of a general inner matrix norm. The approach is called eCauchy possibilistic clustering. Using forms of the general inner matrix norm, this approach can be used in different problems, from classification to regression. A major difference to e.g. TEDA and others is that different norms can be integrated and therefore different shapes and characteristics of the clusters can be induced for different learning problems. Moreover, we suggest a holistic evolving density-based clustering approach integrating not only shape diversity of the clusters, but also methods how to i) add, ii) merge, iii) delete and iv) split clusters in one joint recursive, incremental cluster update method.

This paper is organized as follows: after the introduction, the basics of Cauchy density of a data stream are given, some possible different inner matrix norms are developed and presented which can be used in data stream processing and the recursive computation of cluster covariance matrix is presented. In Section 3 the identification of input–output models is given and in Section 4 the evolving strategies that are used in our approach are presented. In Section 5 the proposed algorithm is illustrated on different practical examples. In 6, the in-depth discussion of the proposed method is presented, giving its strength and showing its weak points with respect to the existing methods from the literature. In the end, some conclusions are made.

2. Similarity measures in streaming data

The similarity metrics between samples coming from streaming data is given by basic Cauchy density and its modifications. The Cauchy density γ_k^j is defined by a suitable kernel over the distances between the current sample $\boldsymbol{z}(k) \in \mathbb{R}^q$ and all the previous samples $\boldsymbol{z}_i^j$ that have already been classified to a particular cluster (jth in this case) [4]:

$$\gamma_k^j = \frac{1}{1 + \frac{\sum_{i=1}^{M^j} (\boldsymbol{z}(k) - \boldsymbol{z}_i^j)^T (\boldsymbol{z}(k) - \boldsymbol{z}_i^j)}{M^j}} \quad j = 1, \ldots, m \tag{1}$$

where M^j is the number of data samples associated with the jth cluster. The density in this form can be seen as an absolute density of the data sample according to the other data samples in the set because of the absolute distances between the samples.

The basic density measure based on the Euclidean distance and the generalization of this distance, which then gives a much more powerful method, is done by introducing a positive definite inner norm matrix of the jth cluster $\boldsymbol{A}^j$ into the distance measure. The Cauchy density becomes relative because the distances are weighted in different directions with different weights. The general expression of the weighted Cauchy density is given by

$$\gamma_k^j = \frac{1}{1 + \frac{\sum_{i=1}^{M^j} (\boldsymbol{z}(k) - \boldsymbol{z}_i^j)^T \boldsymbol{A}^j (\boldsymbol{z}(k) - \boldsymbol{z}_i^j)}{M^j}} \tag{2}$$

Eq. (2) can be transformed into the recursive form in the case of data stream problems as follows (*see* Appendix A):

$$\gamma_k^j = \frac{1}{1 + (\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T \boldsymbol{A}^j (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) + T_A^j} \tag{3}$$

where

$$T_A^j = \frac{(M^j - 1)}{M^j} \operatorname{trace}(\boldsymbol{A}^j \boldsymbol{\Sigma}^j) \tag{4}$$

while

$$\boldsymbol{\Sigma}^j = \frac{1}{M^j - 1} \sum_{i=1}^{M^j} (\boldsymbol{z}_i^j - \boldsymbol{\mu}^j)(\boldsymbol{z}_i^j - \boldsymbol{\mu}^j)^T \tag{5}$$

represents the covariance matrix of the jth cluster which has M^j samples.

It has to be stressed that the definition of the covariance matrix from Eq. (5) is not suitable for analysis of data streams. In such cases recursive calculation of covariance matrix is needed. The algorithm can be found in Appendix B.

The density defined in Eq. (3) has the highest value when the sample $\boldsymbol{z}(k)$ hits the prototype $\boldsymbol{\mu}^j$. In that case the density becomes $1/(1 + T_A^j)$. Since the term T_A^j defined in Eq. (4) depends on the choice of the matrix $\boldsymbol{A}^j$ and the number of input space features, it is hard to predict the maximal value of the density given by Eq. (3) which makes the tuning of the parameters more complicated. Therefore it is preferable to use the normalized density, where the original density from Eq. (3) is normalized with its maximal value $1/(1 + T_A^j)$. The maximal normalized density is equal to 1 for all inner matrix norms and is calculated as follows:

$$\gamma_k^j = \frac{1 + T_A^j}{1 + (\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T \boldsymbol{A}^j (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) + T_A^j} \tag{6}$$

For density calculation, distances with different inner matrix norms can be used to cope with various problems. The normalized densities with different inner matrix norms will therefore be discussed next.

2.1. Identity inner norm

When the identity matrix $\boldsymbol{I}$ is chosen as the inner matrix norm, Euclidean distance is obtained [15] and the density becomes

$$\gamma_k^j = \frac{1 + T_I^j}{1 + (\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) + T_I^j} \tag{7}$$

where $T_I^j = \frac{(M^j-1)}{M^j}\,\mathrm{trace}(\boldsymbol{\Sigma}^j)$. This norm is very basic but can be successfully used in different approximation and classification problems.

2.2. Inverse covariance matrix as inner matrix norm

When the inverse of the covariance matrix $(\boldsymbol{\Sigma}^j)^{-1}$ of the corresponding data cluster with M^j data samples is chosen as the inner matrix norm, the distance is called the Mahalanobis distance [15]. The matrix $\boldsymbol{\Sigma}^j$ from Eq. (5) needs to be invertible to use this matrix norm [7,15]. $\boldsymbol{\Sigma}^j$ is non-singular if and only if the matrix of all centered data samples $\boldsymbol{Z}^j$ has rank q [37]. The matrix $\boldsymbol{Z}^j$ is constructed from centered row samples $(\boldsymbol{z}_i^j - \boldsymbol{\mu}^j)^T$ that form the matrix of dimension $M^j \times q$. It is shown in Appendix A that all the columns in $\boldsymbol{Z}^j$ have zero mean (see Eq. (34)), and therefore at least $q+1$ linearly independent samples are needed for the matrix $\boldsymbol{Z}^j$ (or equivalently $\boldsymbol{\Sigma}^j$) to achieve full rank [37].

Introducing the inverse covariance matrix as inner matrix norm into Eq. (3), the expression for Cauchy relative density based on the Mahalanobis distance is obtained:

$$\gamma_k^j = \frac{1 + \frac{(M^j-1)}{M^j} q}{1 + (\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T (\boldsymbol{\Sigma}^j)^{-1} (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) + \frac{(M^j-1)}{M^j} q} \tag{8}$$

where the term T_A^j (Eq. (4)) in this case equals $\frac{M^j-1}{M^j} q$, since it is easy to show that $\mathrm{trace}(\boldsymbol{A}^j \boldsymbol{\Sigma}^j) = \mathrm{trace}((\boldsymbol{\Sigma}^j)^{-1} \boldsymbol{\Sigma}^j) = q$, and q stands for the number of input space features.

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

$$\boldsymbol{\Sigma}^j = \boldsymbol{P}^j \boldsymbol{\Lambda}^j (\boldsymbol{P}^j)^T \tag{9}$$

with $\boldsymbol{P}^j = \left[\boldsymbol{p}_1^j, \boldsymbol{p}_2^j, \ldots, \boldsymbol{p}_q^j\right]$ and $\boldsymbol{\Lambda}^j$ being the matrix of eigenvectors and eigenvalues of $\boldsymbol{\Sigma}^j$, respectively. Eigenvalues and eigenvectors represent the shape of the corresponding cluster. The Mahalanobis distance can be therefore described in the following form due to (9):

$$(\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T (\boldsymbol{\Sigma}^j)^{-1} (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) = \sum_{r=1}^{q} \frac{\left((\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T \boldsymbol{p}_r^j\right)^2}{\lambda_r} \tag{10}$$

where $\boldsymbol{p}_r^j$ stands for the rth eigenvector and λ_r for the rth eigenvalue of the covariance matrix $\boldsymbol{\Sigma}^j$. It can be seen 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 eigenvectors. The distances are normalized with appropriate eigenvalues. The term $\left((\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T \boldsymbol{p}_r^j\right)$ can be also viewed as a distance to the hyperplane [16] that is defined by rth eigenvector and the cluster center $\boldsymbol{\mu}^j$.

The benefit of using the Mahalanobis distance is that a hyper-ellipsoidally shaped cluster can be described in contrast to the Euclidean distance with which only hyper-spherical shapes can be described [15]. The size and the shape of the hyper-ellipsoid depends on the covariance matrix of the data in the cluster.

A very strong argument to use the Mahalanobis distance or inverse covariance matrix as inner norm is the scalability of the data. The Mahalanobis distance is scale-invariant, whereas in the case of the Euclidean distance it is not the case.

2.3. Inverse of partial covariance matrix as inner matrix norm

In classification problems, the density based on the Mahalanobis distance can be of great use and importance. But when dealing with regression problems the density just in the direction of certain eigenvector(s) is sometimes more important. Latent eigenvectors play an important role in this case. For such cases, partial covariance matrix ${\boldsymbol{\Sigma}_q}^j$ could be used in the computation of density. The inner matrix norm in this case is obtained from $(\boldsymbol{\Sigma}^j)^{-1}$ by only selecting a few eigenvectors and their corresponding eigenvalues. This is obtained by including a square matrix $\boldsymbol{Q}$ of dimension $q \times q$ in Eq. (9):

$$(\boldsymbol{\Sigma}_q^j)^{-1} = \boldsymbol{P}^j (\boldsymbol{\Lambda}^j)^{-1} \boldsymbol{Q} (\boldsymbol{P}^j)^T \tag{11}$$

where $\boldsymbol{Q}$ is composed of zeros and only a few diagonal elements take the value 1. Typically, only the element (q, q) is equal to 1; in this case, only a latent eigenvector is used in the partial covariance matrix.

The latent Mahalanobis distance, i.e. the normalized Mahalanobis distance in the direction of the least important eigenvector $\boldsymbol{p}_q^j$ (the eigenvector with the smallest eigenvalue λ_q) is then obtained as follows:

$$(\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T ({\boldsymbol{\Sigma}_q}^j)^{-1} (\boldsymbol{z}(k) - \boldsymbol{\mu}^j) = \frac{\left((\boldsymbol{z}(k) - \boldsymbol{\mu}^j)^T \boldsymbol{p}_q^j\right)^2}{\lambda_q} \tag{12}$$

The density in the latent direction then becomes

Table 1
An example of different normalized densities.

Type of inner matrix norm	Density at $\boldsymbol{z}(k) = \boldsymbol{\mu} + \alpha_1\sigma_1\boldsymbol{p}_1 + \alpha_2\sigma_2\boldsymbol{p}_2$
Identity matrix	$\gamma_1 = \frac{1+\frac{M-1}{M}(\sigma_1^2+\sigma_2^2)}{1+(\alpha_1\sigma_1^2+\alpha_2\sigma_2^2)+\frac{M-1}{M}(\sigma_1^2+\sigma_2^2)}$
Inverse covariance matrix	$\gamma_2 = \frac{1+\frac{M-1}{M}2}{1+(\alpha_1^2+\alpha_2^2)+\frac{M-1}{M}2}$
Inverse of partial covariance matrix	$\gamma_3 = \frac{1+\frac{M-1}{M}1}{1+\alpha_2^2+\frac{M-1}{M}1}$
Partial eigenvector matrix	$\gamma_4 = \frac{1+\frac{M-1}{M}\sigma_2^2}{1+\alpha_2^2\sigma_2^2+\frac{M-1}{M}\sigma_2^2}$
Normalized partial eigenvectors matrix	$\gamma_5 = \frac{1+\frac{M-1}{M}\frac{\sigma_2^2}{\sigma_l^2}}{1+\alpha_2^2\frac{\sigma_2^2}{\sigma_l^2}+\frac{M-1}{M}\frac{\sigma_2^2}{\sigma_l^2}}$

$$\gamma_k^j = \frac{1+\frac{(M^j-1)}{M^j}1}{1+\frac{\left((\boldsymbol{z}(k)-\boldsymbol{\mu}^j)^T\boldsymbol{p}_q^j\right)^2}{\lambda_q}+\frac{(M^j-1)}{M^j}1} \tag{13}$$

Note the last factor 1 in the denominator of Eq. (13) that has changed with respect to Eq. (8) in which it was q. In general the latent vector together with the cluster center defines a local model.

2.4. Partial eigenvector matrix as inner matrix norm

The eigenvector matrix as an inner matrix norm leads to the absolute density in the direction of the chosen, i.e., latent eigenvector. This matrix norm is defined as follows:

$$\boldsymbol{A}_q^j = \boldsymbol{p}_q^j(\boldsymbol{p}_q^j)^T \tag{14}$$

where $\boldsymbol{p}_q^j$ defines the latent eigenvector of the covariance matrix for the jth cluster with M^j samples. Usually, the direction of the least important eigenvector $\boldsymbol{p}_q$ is chosen because it is the normal vector to the hyperplane spanned with the other $q-1$ eigenvectors that implicitly define the model. The absolute density in the latent direction then becomes

$$\gamma_k^j = \frac{1+\frac{(M^j-1)}{M^j}\lambda_q}{1+\left((\boldsymbol{z}(k)-\boldsymbol{\mu}^j)^T\boldsymbol{p}_q^j\right)^2+\frac{(M^j-1)}{M^j}\lambda_q} \tag{15}$$

Note the last factor λ_q in the denominator of Eq. (15) that has changed with respect to Eqs. (8) and (13).

2.5. Normalized partial eigenvectors matrix as inner matrix norm

Using this inner matrix norm, the partial eigenvectors matrix is normalized with a constant variance σ_l^2, which can be used in the case where the variance of the samples in the latent direction can be roughly estimated. This is a free design parameter that is kept constant throughout the evolving procedure. The matrix norm is defined as follows:

$$\boldsymbol{A}_q^j = \frac{1}{\sigma_l^2}\boldsymbol{p}_q^j(\boldsymbol{p}_q^j)^T \tag{16}$$

where $\boldsymbol{p}_q^j$ defines the latent eigenvector of the covariance matrix for the jth cluster. The density in this case becomes

$$\gamma_k^j = \frac{1+\frac{(M^j-1)}{M^j}\frac{\lambda_q}{\sigma_l^2}}{1+(\boldsymbol{z}(k)-\boldsymbol{\mu}^j)^T\boldsymbol{A}_q^j(\boldsymbol{z}(k)-\boldsymbol{\mu}^j)+\frac{(M^j-1)}{M^j}\frac{\lambda_q}{\sigma_l^2}} \tag{17}$$

Such density calculation could be useful because of its ability to overcome the problems of a learning transient of covariance matrices.

Example – Calculation of previously defined densities

To show the calculation of all five previously defined densities, a simple two dimensional example ($q = 2$) will be given. Assuming the set of data samples with the mean value $\boldsymbol{\mu}$, the covariance matrix $\boldsymbol{\Sigma}$, and a number of samples M. The covariance matrix is decomposed as $\boldsymbol{\Sigma} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^T$, where $\boldsymbol{P} = [\boldsymbol{p}_1\ \boldsymbol{p}_2]$ and $\boldsymbol{D} = \mathrm{diag}(\sigma_1^2, \sigma_2^2)$. The trace of covariance matrix equals $\mathrm{trace}(\boldsymbol{\Sigma}) = \sigma_1^2 + \sigma_2^2$.

Let us define a general data sample so that $\boldsymbol{z}(k) - \boldsymbol{\mu} = \alpha_1\sigma_1\boldsymbol{p}_1 + \alpha_2\sigma_2\boldsymbol{p}_2$, where α_1, and α_2 stands for arbitrary scalars as shown in Fig. 1. The calculation of different densities are given in Table 1. It is shown that by using different inner matrix norms the obtained densities can be absolute or relative according to the size of clusters, i.e. their covariance matrices and their eigenvalues.

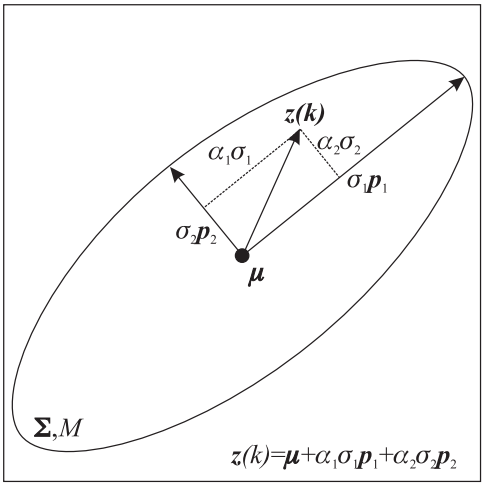

Fig. 1. Calculation of different densities.

3. Identification of the input–output model

The principle of Cauchy density is used to partition the data space and obtain the clusters. The clusters are defined by the center $\boldsymbol{\mu}^j$ and the covariance matrix $\boldsymbol{\Sigma}^j$. In regression problems, local linear models must also be identified, either by recursive least squares method or by defining the hyper-plane that spans over the data. This approach will be shown next.

3.1. Estimation of the local model parameters with hyper-plane

The general nonlinear mapping that maps a compact set from the input space of arbitrary dimension to $\mathbb{R}$ can be approximated by a number of approximators. Quite often, a Takagi–Sugeno model is used. Very well known are the approaches in which the structure of the model is constant and the parameters of the model $\boldsymbol{\theta}_j$ are estimated and adapted on-line. Simultaneous identification of the model parameters $\boldsymbol{\theta}_j$ and partitioning of the input–output data space has also received a great deal of attention in the literature recently. A very simple and fast way of estimating the model is an analytical approach based on the singular value decomposition of the covariance matrices of the clusters.

The measurement vectors are composed of the input vector $\boldsymbol{x}$, which could be of an arbitrary dimension and the corresponding output y, which is a scalar in this case:

$$\boldsymbol{z}^T = \begin{bmatrix} \boldsymbol{x}^T & y \end{bmatrix} \in \mathbb{R}^q, \tag{18}$$

where q is an input–output space dimension. All the information about the data lies in the covariance matrix of the cluster, and the most important part is therefore partitioning, i.e. granulation of the input–output space to obtain the clusters that cover the whole data space adequately. It is assumed that the input–output data in the input–output space lie along the hyper-surface representing the input–output mapping. This assumption is usually true for a huge class of 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 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 $\boldsymbol{n}^j$ to the hyper-surface is orthogonal to the tangential hyper-plane in the center of the cluster $\boldsymbol{\mu}^j$. This tangential hyper-plane represents the local linear model and can be obtained in the implicit equation as follows

$$\left(\boldsymbol{z} - \boldsymbol{\mu}^j\right)^T \boldsymbol{n}^j = 0 \tag{19}$$

The normal vector is defined as

$$\boldsymbol{n}^j = \boldsymbol{p}_r^j \tag{20}$$

where $\boldsymbol{p}_r^j$ denotes the first eigenvector of the covariance matrix $\boldsymbol{\Sigma}^j$ 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, qth eigenvalue should be ≈ 0 and $(q-1)$th eigenvalue should reflect the noise. Therefore for this case $r = q - 1$.

In the case of regression problems, the regressors are usually very carefully chosen, meaning that the regressors 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 $\boldsymbol{n}^j$ is equal to the latent eigenvector $\boldsymbol{p}_q^j$.

Eq. (19) represents the implicit expression of the hyper-plane of the local linear model that models the data of the jth cluster. The output of the jth local linear model can be obtained from the explicit form of the model that is given as

$$y^j = -\frac{(x_1 - \mu_1^j)p_{q,1}^j + \cdots + (x_{q-1} - \mu_{q-1}^j)p_{q,q-1}^j}{p_{q,q}^j} + \mu_q^j \tag{21}$$

where $\boldsymbol{x} = \begin{bmatrix} x_1 & \dots & x_{q-1} \end{bmatrix}^T$ is the input vector, $\boldsymbol{p}_q^j = \begin{bmatrix} p_{q,1}^j & \dots & p_{q,q}^j \end{bmatrix}^T$ and $\boldsymbol{\mu}^j = \begin{bmatrix} \mu_1^j & \dots & \mu_q^j \end{bmatrix}^T$ are the normal vector and the center vector, respectively, of the jth hyper-plane. In a more compact form, the model output can be represented as follows

$$y^j = -\frac{1}{p_{q,q}^j}(\boldsymbol{x} - \hat{\boldsymbol{\mu}}^j)^T \hat{\boldsymbol{p}}_q^j + \mu_q^j \tag{22}$$

where $\hat{\boldsymbol{\mu}}^j = \begin{bmatrix} \mu_1^j & \dots & \mu_{q-1}^j \end{bmatrix}^T$ consists of the first $q-1$ elements of the vector $\boldsymbol{\mu}^j$ whereas $\hat{\boldsymbol{p}}_q^j = \begin{bmatrix} p_{q,1}^j & \dots & p_{q,q-1}^j \end{bmatrix}^T$ includes the first $q-1$ coefficients of the normal vector $\boldsymbol{p}_q^j$.

3.2. Fuzzification of local models

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 density, 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:

$$y(k) = \frac{\sum_{j=1}^m \gamma_k^j y^j}{\sum_{j=1}^m \gamma_k^j} \tag{23}$$

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

$$\hat{\boldsymbol{z}}^T = \begin{bmatrix} \boldsymbol{x}^T & y^j \end{bmatrix} \tag{24}$$

The typicalities can be replaced by the membership values obtained from the following equation:

$$\gamma_k^j = e^{-\frac{1}{2}(\hat{\boldsymbol{z}}(k) - \boldsymbol{\mu}^j)^T (\boldsymbol{\Sigma}^j)^{-1} (\hat{\boldsymbol{z}}(k) - \boldsymbol{\mu}^j)} \tag{25}$$

When using the method for classification problems, the size of the output vector is one, and the output represents the label. The label is assigned when a cluster is created. The output of the model is calculated without fuzzification in the case of the classification model, i.e. the output is then the label of the cluster with the highest value of density measure.

4. Evolving strategies

The evolving strategies strongly depend on the investigated problem. The problems that are commonly treated are the classification problems in which the data from the stream are classified into the different classes and the regression problems where the identification of a process model is treated. In the case of classification problems, the data are spanned in all directions and usually the rank of data covariance matrix equals the number of variables. In the case of regression problems, the data can be divided into input and output variables (in our case, only one output is used) and the main goal is to find the inner data relation between the inputs and the output. The data lie along a hyper-plane with a zero or low variability in one or more dimensions.

The used evolving strategy also strongly depends on the nature of consecutive samples in the data stream. The problems can be basically divided into two different classes. In the first class are the problems in which the samples come from different processes randomly. Dealing with such problems, all evolving mechanisms of adding, merging and removing the clusters are necessary. The problems can also be solved by semi-evolving algorithms, which introduce the buffer for unclassified samples, which is processed when certain conditions are met. In the second class, we have the processes in which the consecutive samples come from processes in a certain order; they fall into the neighboring clusters and have a continuous and smooth behavior. In this cases, the clusters arise mainly because of the nonlinearity in the process that generates the data stream. A new cluster is added directly when certain conditions are fulfilled.

Next, the basic evolving mechanisms used in this paper are discussed, i.e. adding, removing, and merging the clusters.

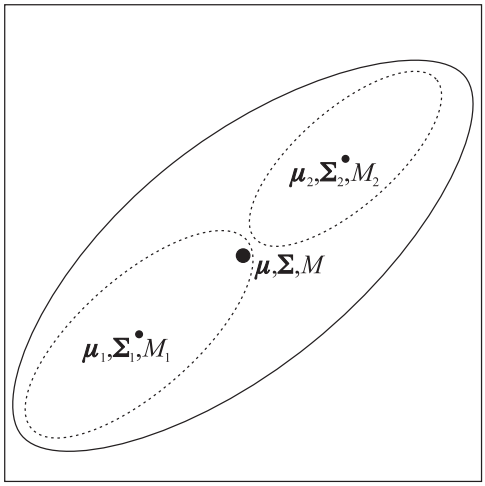

Fig. 2. Basic principle of merging clusters.

4.1. Adding clusters

When dealing with classification problems the use of direct evolving adding method is the most common. This means that each sample is either added to one of the existing clusters or a new cluster is initiated. The new cluster is added if

$$\max_j \gamma_k^j < \Gamma_{max} \tag{26}$$

where Γ_{max} stands for the density threshold [4]. When a new cluster is added, the number of clusters m is incremented, the number of elements in the new cluster is set to 1 (the current sample), and the center and the covariance matrix are initialized as $\boldsymbol{\mu}^j = \boldsymbol{z}(k)$ and $\boldsymbol{\Sigma}^j = \alpha\mathbf{I}$ (α is a "large number" as mentioned before).

The semi-evolving adding method involves the data buffer. If the samples, which are not typical for the existing clusters and also do not lie far away from each other, are found in the buffer, a new cluster is initiated with these data. The sample is added to the buffer if an additional condition to the condition (26) is satisfied:

$$\max_j \gamma_k^j > \Gamma_{min} \tag{27}$$

This means that in the indirect case the lower density threshold Γ_{min} is also defined to eliminate the samples that are very untypical. When the buffer is full, a new cluster center and the covariance matrix are calculated from the data in the buffer. The size of the buffer depends on the data dimension and is defined by the user.

4.2. Removing clusters

The method used to remove clusters strongly depends on the adding method used. When the indirect adding method is used, the clusters are usually well defined, and there is very rarely the need for removing already defined clusters. If the direct method is used, this is not the case. It can happen that a cluster is defined with just one or two samples, and therefore, in regard to the problem nature, the clusters with the insufficient number of samples ($M^j < q$) are removed [12].

4.3. Merging clusters

Dealing with evolving methods, the mechanism of merging is always unavoidable [22]. There are many different principles of detecting the clusters which should be merged and also a lot of principles of merging. First, it is necessary to determine which clusters should be merged. The procedure always tries to find the closest clusters, in predefined metrics, to be merged. This is then repeated until the so-called close clusters exist. The basic idea of merging clusters is shown in Fig. 2. In our paper, the merging of clusters is not of the main interest and, therefore, we used the known method which is proposed in [29] where the idea of homogeneity is used.

The above-mentioned merging procedure is more suitable for clustering problems. For classification and regression problems a more suitable supervised merging mechanism is presented in [12]. In the case of supervised merging, the algorithm searches for pairs of neighborhood clusters. The algorithm then checks if the difference between the local models' parameters are below a certain threshold. If this is true then the clusters are merged. In the case of classification, the labels of clusters must be the same.

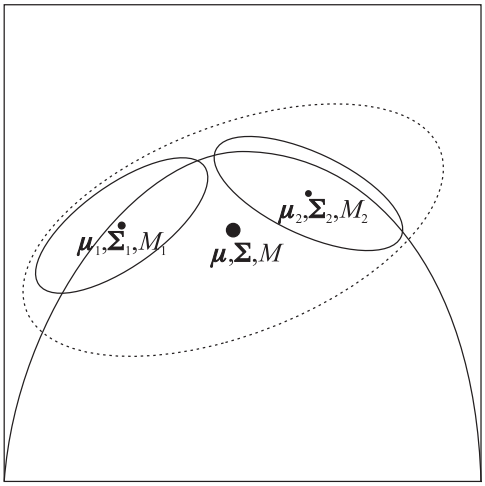

Fig. 3. Basic principle of splitting a cluster.

4.4. Splitting clusters

The splitting of clusters is usually implemented in evolving algorithms to ensure that clusters with high prediction error or clusters including different labels in classification case are split into two clusters to ensure better overall model performance. The basic idea of splitting in the case or regression is shown in Fig. 3. When a newly created cluster receives enough support samples (e.g. 30 samples), sub-clusters are created by splitting the parent cluster by moving the cluster centers in the direction of the maximal eigenvector [32,43]. The algorithm then checks the approximation error in the case of new sub-clusters and compare it with approximation in the case of parent cluster. If splitting gives an improvement which is above a certain threshold one of the sub-clusters replaces the parent cluster, and the other is added as a new cluster.

4.5. Computational complexity

Computational complexity is also one of important issues of the algorithm because it should be performed in real-time on the data stream. The complexity is given in O-term notation for indicating the number of flops. The algorithm of the recursive Cauchy algorithm with the Mahalanobis distance has a complexity of $O(mq^2)$ where q stands for the dimensionality of the feature space and m for the number of clusters, recursive Cauchy with Euclidean distance has complexity of $O(mq)$, recursive Cauchy with partial covariance $O(mq)$, recursive Cauchy with partial eigenvectors $O(mq^2)$, recursive mean $O(q)$, because only the winning cluster is adapted, recursive covariance matrix $O(q^2)$, adding clusters $O(m)$, and the complexity of $O(q) + O(q^2) + O(q^3)$ for merging the clusters.

5. Validation of the eCauchy algorithm on practical examples

5.1. Clustering problem example

To show the use of eCauchy method for clustering problems, the data that are shown as the time courses of variable x_1 and x_2 in Fig. 4 are used. The measured samples come from different processes and this results in data that belong to different clusters. Although the clusters can be of different forms, the most robust solution can be found by using the Euclidean distance, or the identity matrix as the inner norm, $\boldsymbol{A}^j = \boldsymbol{I},\ j = 1, \ldots, m$. The Mahalanobis distance suffers from singularity problems according to the inverse of covariance matrix. The density threshold to add a new cluster is defined as $\Gamma_{max} = 0.5$. The results of clustering are depicted in Fig. 5. Some outliers are also present in the data, which do not influence the partitioning. The time course of cluster centers for different clusters as they appear during the evolving procedure are given in Fig. 6.

The clusters in Fig. 5 can also be merged if very similar clusters according to the chosen merging criteria are encountered. The merged clusters are given in Fig. 7. In the figure, the probability areas of so-called 3σ are also plotted. This means that this area covers 99.7% of all data samples.

5.2. Function approximation – Example I

As an example of function regression, the function

$$x_2 = 4\sin(0.05x_1)\exp(-0.05x_1)$$

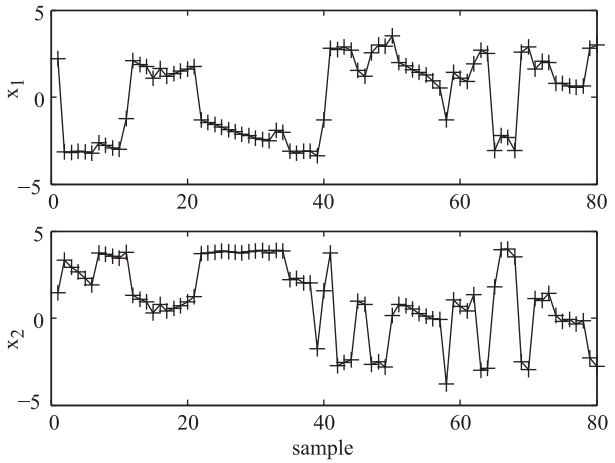

Fig. 4. The time course of variables x_1 and x_2 where the measured samples come from different processes (defined by different clusters).

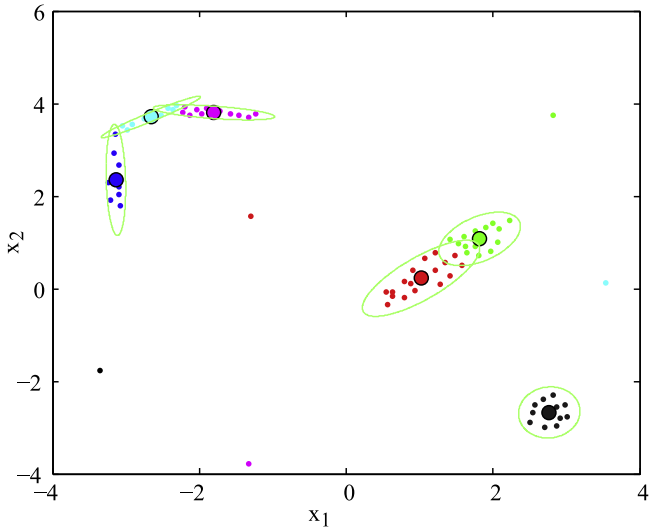

Fig. 5. The clusters in the case of eCauchy clustering without merging the clusters (with probability area of 3σ).

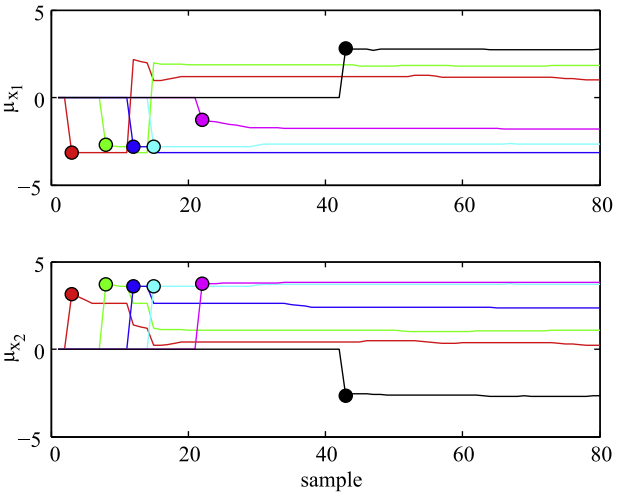

Fig. 6. The time course of cluster centers for different clusters in the case of classification.

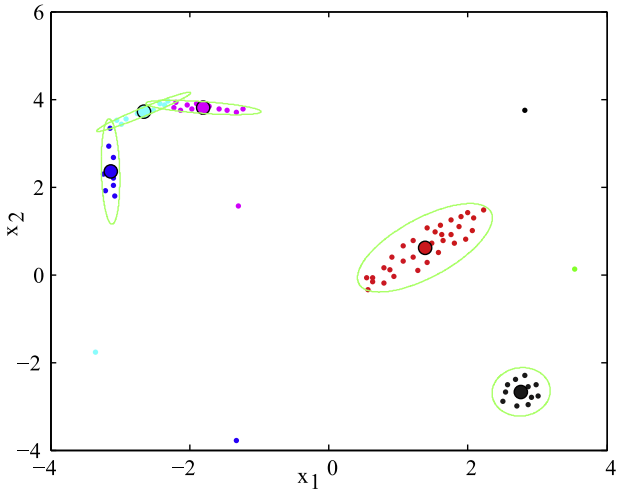

Fig. 7. The clusters in the case of eCauchy clustering after merging the clusters (with probability area of 3σ).

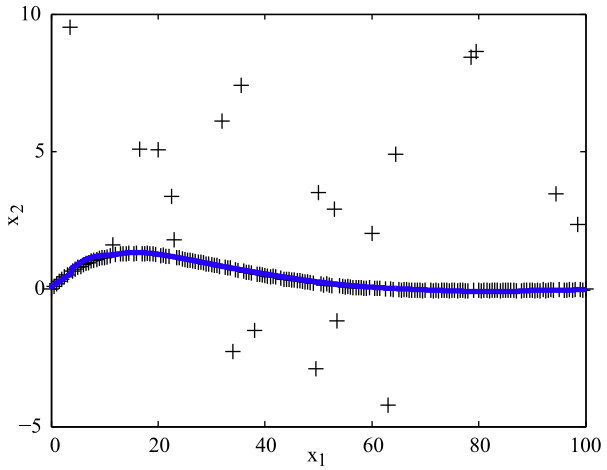

Fig. 8. The approximation of the function.

is used, where $x_1 = kT_s$, with $k = 0, \ldots, 200$ and $T_s = 0.5$. The noise with characteristics $\mathcal{N}(0, 0.01)$ was added to the output of the function together with some outliers. The whole dataset is shown by + markers in Fig. 8. As can be noticed, a large number of outliers is present.

A semi-evolving strategy was used to identify the data stream on-line. The lower and the upper boundary values for density (for putting a sample in the buffer) are defined as $\Gamma_{min} = 0.4$ and $\Gamma_{max} = 0.5$. The buffer with the size of 3 samples is utilised. In this case the normalized partial eigenvectors matrix with variance $\sigma_l^2 = 0.09$ is used as inner matrix norm.

In Fig. 8, the function approximation obtained by the proposed algorithm is given in blue. It is shown that the outliers are completely ignored, and the approximation does not depend on them. The approximation is very robust and deterministic. This cannot be obtained by using the identity matrix as the inner norm.

The clusters with the centers and the probability region of 3σ for the function approximation problem are shown in Fig. 9. The cluster centers together with the covariance matrices define the local linear models that approximate the function $x_2 = f(x_1)$. The local models and centers together with samples that belong to a certain cluster are presented in Fig. 10. In Fig. 11, the time course of cluster centers for different clusters in the case of function approximation is given, for both variables, x_1 and x_2.

5.3. Example of mapping on real laser range finder data

The laser range finder (LRF) or LIDAR, which stands for Light Detection and Ranging, is a remote sensor that uses light in the form of a pulsed laser to measure variable distances to the obstacles from which the light is reflected. In the case of 2D laser range finder the data stream of reflected points belongs to consecutive laser rays that are sent to the environment from a starting angle and they increment equidistantly to a final angle, and they lie in the same plane. The LRF is a very

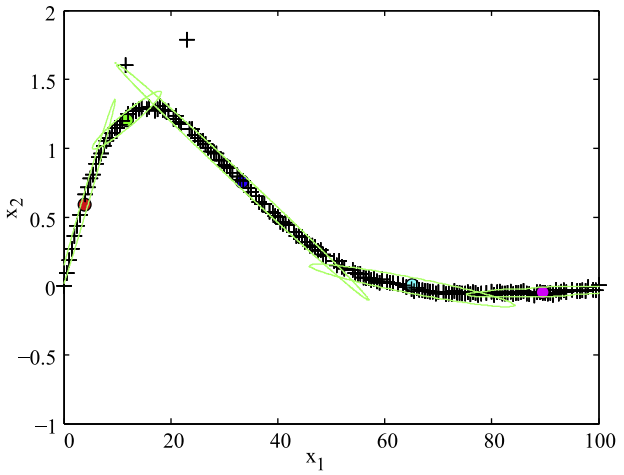

Fig. 9. The clusters with the centers and the probability region of 3σ for the function approximation problem.

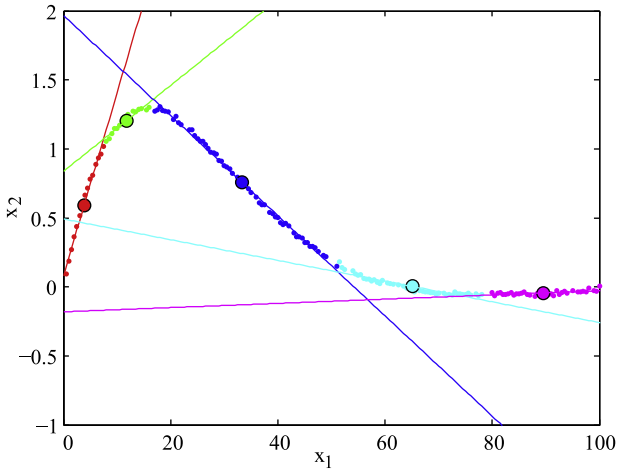

Fig. 10. The cluster centers with local linear models in the case of the function approximation problem.

popular sensor for mapping purposes in mobile robotics. This is because of good coverage, dense information, high accuracy, and a high sampling rate. The information from LRF can be used for localization purposes, map building, or simultaneous localization and mapping (SLAM). The robot pose can be estimated by using the data from the LRF and comparison with the given map of the environment. The comparison is usually based on 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 phase of data clustering to find the points that can be described by a line and the phase of a line-parameter estimation. In dealing with the points from a 2D LRF, the first step of the procedure can be simplified, because the clusters always consist of consecutive points, i.e. the points are sorted in the data stream.

The proposed eCauchy clustering involves a procedure that can cluster the data and estimate the parameters in a single compact, procedure. The experimental results are obtained on a 2D scans from a SICK LMS200, which is of the LiDAR class, and therefore provide the range information for a 2D map. The use of the proposed method with some modifications can be seen in more detail in [23].

When we are trying to find linear features in the data space, the most appropriate choice of inner matrix norm is the normalized partial eigenvectors matrix. In our case, the variance $\sigma_l^2 = 0.01$ was used to normalize the density in the latent direction. To avoid overly large clusters in the direction of the main eigenvector, the density was separately calculated in the main direction with $\sigma_m^2 = 1$. The sample, therefore, belongs to a particular cluster if the density in the latent direction is greater than $\Gamma_{max}^l = 0.48$ and in the main direction greater then $\Gamma_{max}^m = 0.40$; otherwise a new cluster is initialized.

In Fig. 12, the normalized real LRF data are plotted. The detected clusters together with centers are given in Fig. 13. In Fig. 14, the detected centers and cluster centers with probability area of 3σ are given. In Fig. 15, the detected clusters and estimated objects in the space are presented with line segments.

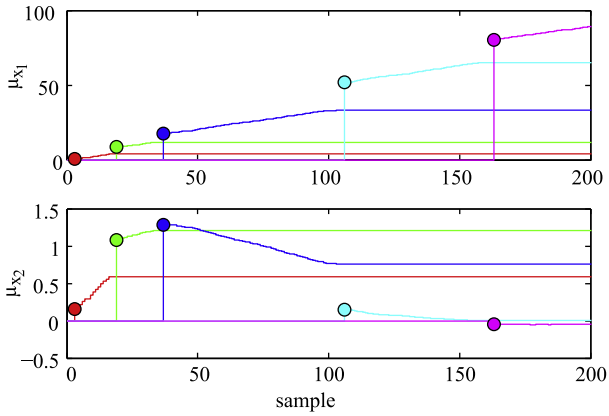

Fig. 11. The time course of cluster centers for different clusters in the case of the function approximation.

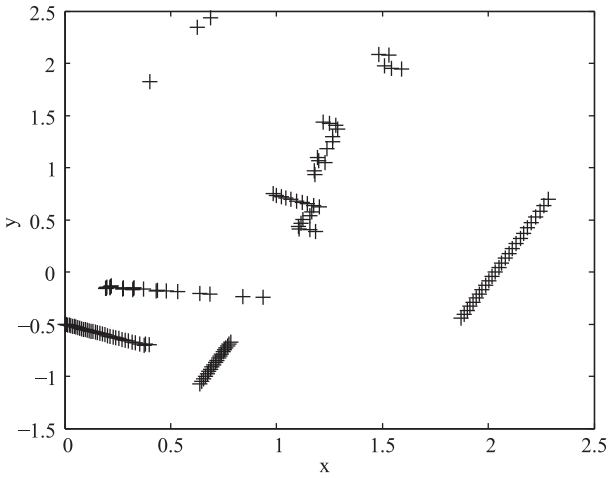

Fig. 12. The normalized raw data from the LRF.

5.4. Benchmark on Mackey–Glass time series

The method was also tested on a Mackey–Glass (M–G) time series prediction. The chaotic time series is generated from the M–G differential delay equation defined by the following equation:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{28}$$

The aim is to use past values of x to predict some future value of x. We assume $x(0)$=1.2, τ=17 and the value of the signal is predicted 85 steps ahead, based on the values of the signal at the current moment, 6, 12, and 18 steps back.

$$Output:[x(t+85)] \tag{29}$$

$$Input:[x(t-18)\ x(t-12)\ x(t-6)\ x(t)] \tag{30}$$

In this example, 3000 data points at $t\in$ [201, 3200] were created for the training, and 500 at $t\in$ [5001, 5500] were created for the testing, the same as in [21]. The fuzzy model evolved during the first 3000 data samples and then this model was used to predict the output for the 500 testing samples. The results from other methods are taken from [12,21] and are shown in Table 2 in which notation eC_e represents the eCauchy method with the Euclidean distance and eC_m with the Mahalanobis distance. The settings for the Euclidean distance were the following: $\Gamma_{max}=0.95$, $\Gamma_{min}=0.001$. For the Mahalanobis distance: $\Gamma_{max}=0.6$, $\Gamma_{min}=0.001$, and $\alpha=1$ for the result with 8 clusters; for the result with 41 clusters, the parameters were set as $\Gamma_{max}=0.6$, $\Gamma_{min}=0.5$, and $\alpha=0.05$; and for the results with 61 clusters, $\Gamma_{max}=0.55$, $\Gamma_{min}=0.1$, and $\alpha=0.02$. In all cases, the minimal number of samples that the clusters must have was set to 10, and the size of the buffer was 30 samples. The local models were identified by least squares.

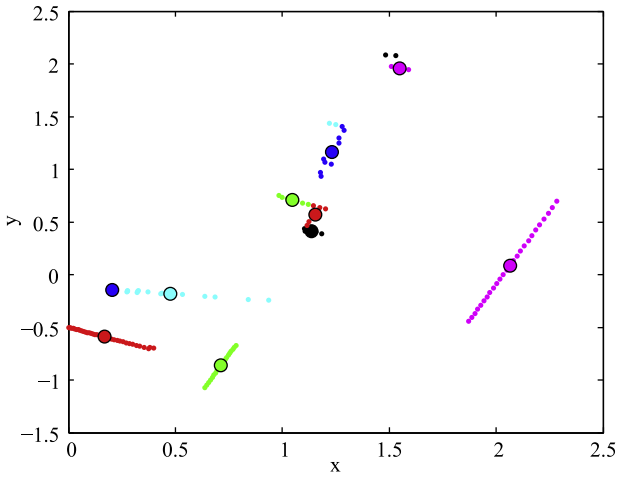

Fig. 13. The detected clusters with the centers.

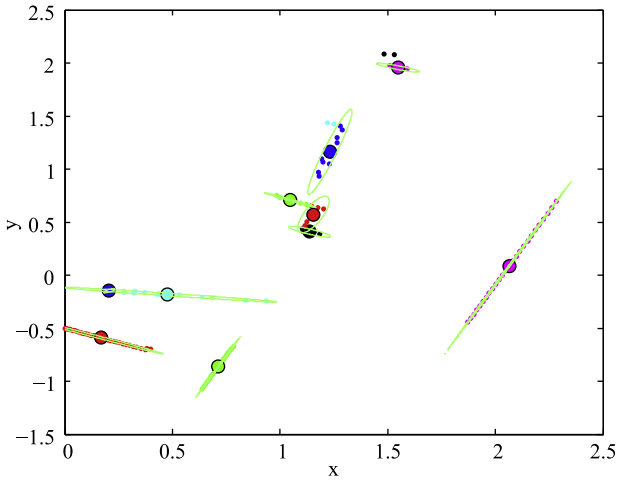

Fig. 14. The detected clusters with centers and the contour with probability area of 3σ.

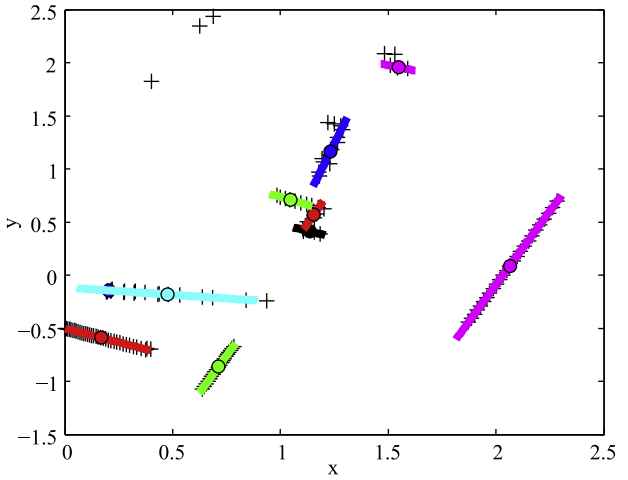

Fig. 15. The detected clusters with centers and lines.

Table 2
Results for Mackey–Glass time series.

Method	Rules	RMSE
DENFIS [21]	58	0.0628
DENFIS [21]	27	0.0920
exTS	10	0.0754
eTS+	10	0.0892
eTS [2]	113	0.0217
rGK [12]	58	0.0481
rGK [12]	10	0.0862
rFCM [10]	10	0.1039
rFCM [10]	58	0.0702
rFCM [10]	100	0.0285
eFuMo [12]	21	0.0753
eFuMo [12]	41	0.0316
eFuMo [12]	68	0.0224
RAN	113	0.0854
ESOM	114	0.0729
EFuNN	193	0.0913
eC_e	24	0.0374
eC_m, $\Gamma_{max} = 0.6,\ \Gamma_{min} = 0.001$	8	0.0989
eC_m, $\Gamma_{max} = 0.6,\ \Gamma_{min} = 0.5$	41	0.0297
eC_m, $\Gamma_{max} = 0.55,\ \Gamma_{min} = 0.1$	61	0.0243
eC_m-noise	12	0.0866

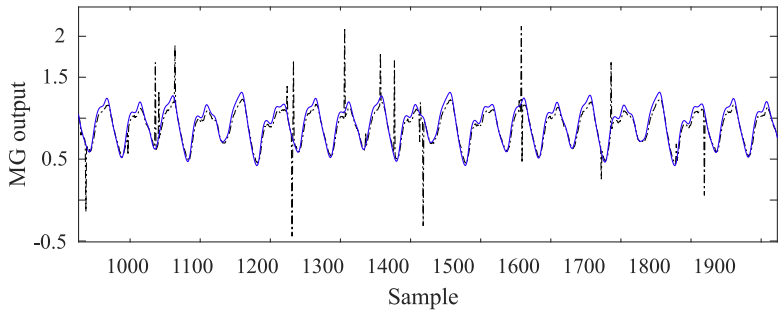

Fig. 16. MG noise data compared to non-noise data.

We can see that the presented method gives results that are very comparable to the other methods, especially to those with a reasonable number of clusters. Even with the added noise (eC_m-noise), the eCauchy method produces reasonably good results. The result eC_m-noise represents the accuracy of the obtained model on a test dataset. The learning dataset was corrupted with noise and outliers. On each input and output variable, a Gaussian noise of variance 0.01 was added independently. Additionally, 2% of the data were corrupted with the Gaussian noise of variance 1. The comparison of the corrupted and original output is shown in Fig. 16.

5.5. Function approximation – Example II

In this subsection we treat the benchmark problem defined in [46]. A dynamical system is given by the difference equation:

$$y(k+1) = \frac{ay(k)}{(1+by(k)^2)} + cu(k)^3 \tag{31}$$

where $y(k)$ and $u(k)$ are the system output and system input, respectively. The parameters a, b, and c were all set to 1. The input to the model is defined as $u(k) = \sin\left(2\pi \frac{k}{100}\right)$. As in [46], the learning set included 50,000 data points, and 200 data points were used for the testing. In Table 3, the results are given for the modeling of the dynamical system given by Eq. (31). The parameters for the Euclidean distance were set as $\Gamma_{max} = \Gamma_{min} = 0.85$; and for the Mahalanobis distance as $\Gamma_{max} = \Gamma_{min} = 0.7$ and $\alpha = 0.02$. In both cases, the minimal number of samples in a cluster was set to 10, and the size of buffer was 30 samples. The local models were identified by least squares. In this case, it can be observed that the Mahalanobis distance gives much better results than the Euclidean distance. The accuracy of the method is not as good as, for example, with the eFuMo method but is comparable to the methods SAFIS, SOFIN, and eTS.

Table 3
Results for the modeling of the dynamical system given by Eq. (31).

Method	Clusters	RMSE
SAFIS [38]	8	0.0116
MRAN [19]	10	0.0129
RANEKF [47]	11	0.0184
simpl_eTS	18	0.0122
eTS [2]	19	0.0082
SONFIN [18]	10	0.013
SAFIN [46]	13	0.007
eFuMo [12]	12	0.0035
eC_e	17	0.0224
eC_m	11	0.0119

Table 4
Results for regression problem under different noise levels.

Noise std.	Clusters mean	RMSE mean	RMSE std. dev.
0.001	9.36	0.0072	0.0011
0.005	9.84	0.0057	0.0011
0.01	11.36	0.0064	0.00078
0.03	14.74	0.0107	0.0021
0.06	16.16	0.0152	0.00089

Table 5
Results for regression problem with different number of outliered data.

% of outliers	Clusters mean	RMSE mean	RMSE std. dev.
5%	9.86	0.008	0.0046
10%	10.22	0.0081	0.0031
20%	10.12	0.0093	0.0027
30%	10.56	0.01	0.0051
50%	9.3	0.015	0.0052

Noise performance study for function approximation example II

The presented methodology performance was evaluated on a test using the same system as in function approximation example II, but with the learning data corrupted with Gaussian noise of different variances. For the learning set, 10,000 data samples were used that were corrupted by Gaussian noise of the following standard deviations: 0.001, 0.005, 0.01, 0.03, and 0.06. The data were normalized before the noise was added. The RMSE was evaluated on a normalized non-noisy dataset of 200 samples. The Γ_{max} and Γ_{min} were set to 0.45 for all tests, and the Mahalanobis distance was used. The results are presented in Table 4. For each noise level, 50 tests were made. The table shows the mean number of clusters generated, mean RMSE value obtained and standard deviation of the RMSE value. For the described experimental setup, the RMSE value on a original data was 0.0077, and 8 clusters were generated.

The method was further tested for outliers with the same setup. Again, the learning dataset was corrupted by Gaussian noise. In this case the standard deviation of the noise was 0.3. However, in this case only a certain percentage (5%, 10%, 20%, 30%, and 50%, respectively) of the data was corrupted. The positions of the corrupted data were chosen randomly. The results are presented in Table 5.

5.6. Box–Jenkins example

The results for the Box–Jenkins gas-furnace benchmark data are presented. Two different experimental setups are treated. In the first case, all the data are used for testing and validation as described in [40]. The error is calculated for the next sample (one-step ahead prediction), and the results are summarized in Table 6. Next, as described in [41], the first 200 data points are used for the learning and the remaining 90 data points are used for the validation. The results are given in Table 7. For both experiments, the design parameters were the same: for the Euclidean distance $\Gamma_{max} = 0.9$ and $\Gamma_{min} = 0.1$; and for the Mahalanobis distance $\Gamma_{max} = 0.5$, $\Gamma_{min} = 0.48$, and $\alpha = 1$. The minimal number of samples in the cluster was set to 5, and the buffer size to 10 samples.

In this example, the presented method gave very good results in comparison to other methods. In the first experimental setup, only DENFIS was better but with much more clusters; and in the second experimental setup only eFuMo was better but with one cluster more.

Table 6
Results for Box–Jenkins gas furnace data in which all the data are used for training and validation.

Method	Clusters	RMSE
DENFIS [21]	12	0.0190
eFuMo [12]	3	0.0337
ANYA [5]	7	0.0393
ELM [17]	20	0.0232
eHFN [39]	7	0.0245
eNNEL [40]	7	0.0354
eC_e	4	0.0258
eC_m	3	0.0257

Table 7
Results for Box–Jenkins gas-furnace data in which 200 data points are used for training.

Method	Clusters	RMSE
eTS [2]	5	0.0490
simpl_eTS	3	0.0485
SOFNN [25]	4	0.0480
SOFMLS [41]	5	0.0474
eFuMo [12]	4	0.0433
eC_e	3	0.0457
eC_m	3	0.0444

Table 8
Results for Rolling-Mill example.

Method	MAE	Clusters	#AE > 20
Analytical	7.84	1	259
Static fuzzy models	6.76	N/A	176
OS-ELM	6.31	30	175
FAOS-PFNN	5.53	24	156
FLEXFIS (conv. EFS)	5.41	N/A	159
FLEXFIS (conv. EFS)	4.65	N/A	68
Gen-Smart-EFS	4.28	24	45
Gen-Smart-EFS	4.28	24	45
Gen-Smart-EFS	4.28	18	47
Gen-Smart-EFS	4.34	12	53
eC_m	5.17	2	257
eC_e	5.19	2	157
eC_e	5.09	8	129

5.7. Rolling-Mill example

In this example, a Rolling-Mill dataset as presented in [31] is used. The goal is to predict the resistance value of a steel plate in a Rolling-Mill; 11 measurement variables are recorded per second in addition to the resistance value. Two datasets with 6503 and 6652 samples, respectively, recorded at two different points of time (2 months apart) have been recorded and stored in the same order as they appeared on-line. The first dataset was used only for learning the model, whereas with the second dataset the one-step-ahead predictions were made before adapting the model (as in [31]). Table 8 shows the results in terms of the one-step-ahead mean absolute error (MAE) on the test dataset. The last column indicates the extreme deviation values (more than 20 units of absolute error). The settings for the Euclidean distance that produced two clusters were the following: $\Gamma_{max} = \Gamma_{min} = 0.55$. The minimal number of samples was set to 50 and the buffer size to 100 samples. The same settings were used for the Mahalanobis distance with the difference in Γ_{max} and Γ_{min} values that were set to 0.25. For the Euclidean distance with 8 clusters, the buffer size was set to 20 samples and minimal number of samples to 5; Γ_{max} and Γ_{min} remained the same. In this case, the Euclidean distance produced better results than Mahalanobis norm. Although the average mean error is lower, there are more predictions with high absolute error (last column of Table 8).

5.8. Classification of Iris data

In this example, the standard benchmark test was made on the classification of the Iris data. The Iris dataset is a well-known dataset including four features and three classes and enjoys great popularity in the pattern recognition community

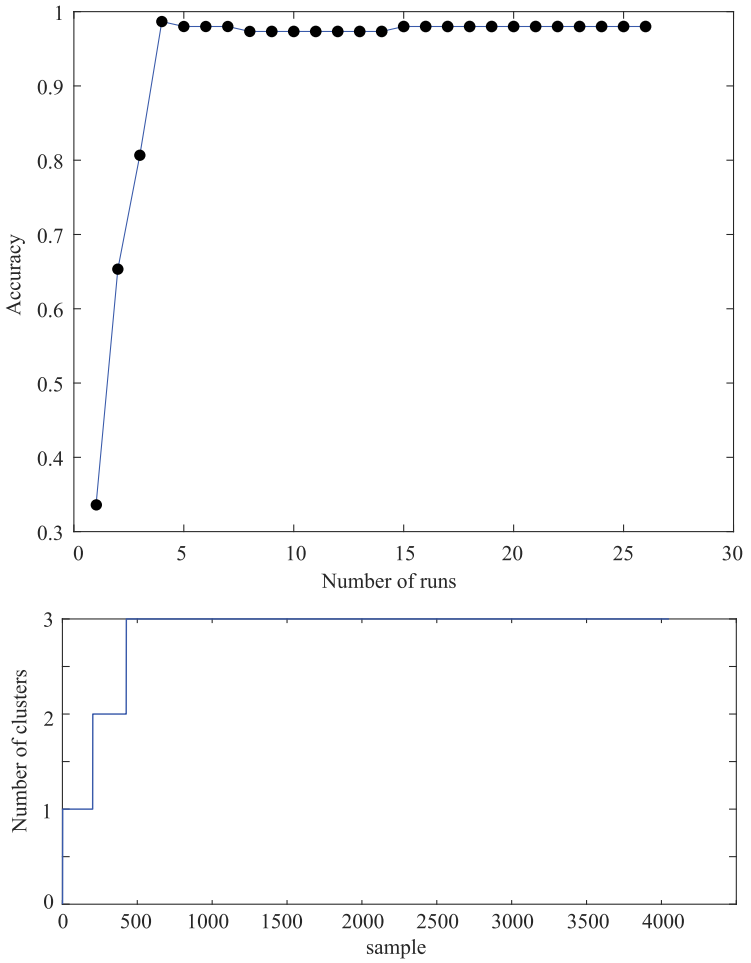

Fig. 17. Iris classification accuracy over multiple runs and cluster evolution using Mahalanobis distance.

for comparing classification approaches. The goal is to recognize the species of flowers (setosa, versicolor, and virginica) based on length and width criteria of their blossoms. The data consists of 150 samples. The algorithm was learned and tested on-line, meaning that first the classification was made, and then the data sample was used to update the classifier. The classification error was calculated for 150 samples. The data was fed to the algorithm multiple times. Fig. 17 shows the improvement of the classifier over multiple runs and the evolution of clusters. The parameters were set to $\Gamma_{max} = 0.6$, $\Gamma_{min} = 0.58$, $\alpha = 0.5$, buffer size 20 and minimal samples 10. The accuracy achieved by the classifier at the end of the experiment was 0.98, which is an accuracy that can be easily compared with other classifiers' accuracies.

Noise performance study for classification of Iris data

The noise performance of the methodology was tested on the Iris dataset. The testing set and the learning set were comprised of the same data. However, the learning set was corrupted by noise and outliers in the same manner as with the regression test. For learning, the method performed 20 sweeps through the learning data. The learning then stopped, followed by testing on 150 test samples. The method settings were the same for all tests: $\Gamma_{max} = \Gamma_{min} = 0.45$, and the Mahalanobis distance was used with $\alpha = 0.5$. Table 9 represents the average generated clusters, mean accuracy and standard deviation of accuracy for different noise levels. The table also represents the minimal, mean, and maximal obtained purity of clusters. The purity of a cluster is calculated as:

$$p_i = \frac{n_{ii}}{n_i} \tag{32}$$

where n_i is the number of data assigned to cluster i while n_{ii} is the number of data assigned to cluster i with the correct label. For non-noisy data the obtained accuracy was 0.97, and 3 clusters were created. The purity for two clusters was 1, and for the third cluster 0.9091. The results for tests on outliers are presented in Table 10.

Table 9
Results for Iris classification problem under different noise levels.

Noise std. dev.	Clusters mean	Min/Mean/Max Cluster Purity	Mean Accuracy	Accuracy std. dev.
0.001	3	0.9091/0.9806/1	0.9795	0.0023
0.005	3	0.9091/0.9712/1	0.9689	0.0042
0.01	3	0.9091/0.9708/1	0.9684	0.0035
0.03	3	0.8929/0.9717/1	0.9697	0.0062
0.06	3	0.0152	0.9074/0.9693/1	0.0036

Table 10
Results for Iris classification problem with different percentages of outliers.

% of outliers	Clusters mean	Min/Mean/Max cluster purity	Mean accuracy	Accuracy std. dev.
5%	2.92	0.5/0.9611/1	0.9476	0.0839
10%	2.82	0.5/0.9403/1	0.9113	0.1159
20%	2.72	0.5/0.9237/1	0.882	0.1357
30%	2.5	0.5/0.8784/1	0.8137	0.1492
50%	2.18	0.5/0.7989/1	0.7161	0.1082

Table 11
Results for digit recognition.

Method	Clusters	Accuracy
FLEXFIS-Class SM	16	0.8877
FLEXFIS-Class MM	61	0.9623
CART	NA	0.9768
k-NN	NA	0.9740
eC_e	76	0.8759
eC_m	71	0.9380

5.9. Pen-based recognition of handwritten digits

This dataset from the UCI repository was created by collecting 250 samples from 44 writers, which were generated with the use of a pressure sensitive tablet with an integrated LCD display and a cordless stylus. The dataset contains 16 features, 7494 training data samples, and 3498 test data samples, containing ten different classes (for the ten digits) which are almost equally distributed in both training and test dataset. Principally, this dataset is an off-line batch dataset. However, we simulate it as an on-line pseudo-stream by performing a loading of data samples and evolve the fuzzy classifiers with each new incoming point separately. The obtained results are compared with the results obtained in [27] in Table 11. Fig. 18 shows the evolution of clusters (bottom graph) and accuracy of the classifier depending on the number of processed samples from the learning set (top graph). The end results obtained by the presented method are not as good as the results of the other four methods. However, the graph in Fig. 18 shows that the top performance using the Mahalanobis distance was around 0.96 with 63 clusters, which is similar to the performance of the *FLEXFIS – ClassMM* method.

5.10. Egg classification

In this example, the goal is to classify the eggs into two classes: broken eggs and not broken eggs. The inspection is made using image analysis. For each egg, an image is taken and different characteristics are calculated. In total 19 inputs are used to classify the eggs [28]. The labels were provided by quality control experts. A set of 2895 samples was used for training, and 2302 samples were used for the evaluation of the classifier performance. The results are compared to results reported in [28] and are given in Table 12. The information about the number of clusters was not available for the compared methods. The settings for the minimal number of samples and buffer size were the same for both distances and were set to 5 and 10 samples, respectively. For both distances, parameters Γ_{max} and Γ_{min} were set to 0.6 and 0.58, respectively. The parameter in case of the Mahalanobis distance was set as $\alpha = 0.5$.

6. Discussion

This paper presents a common evolving framework for on-line clustering, classification, and regression tasks based on Cauchy density and its modifications. The paper also presents different inner matrix norms. The results show that the inverse of the partial covariance matrix and partial eigenvectors matrix can sometimes be beneficial, especially in a low dimension problem and when the regressors are linearly independent. For the general case, it seems that the use of either the Mahalanobis or Euclidean distance is more advisable since we could not obtain good enough results for benchmark problems

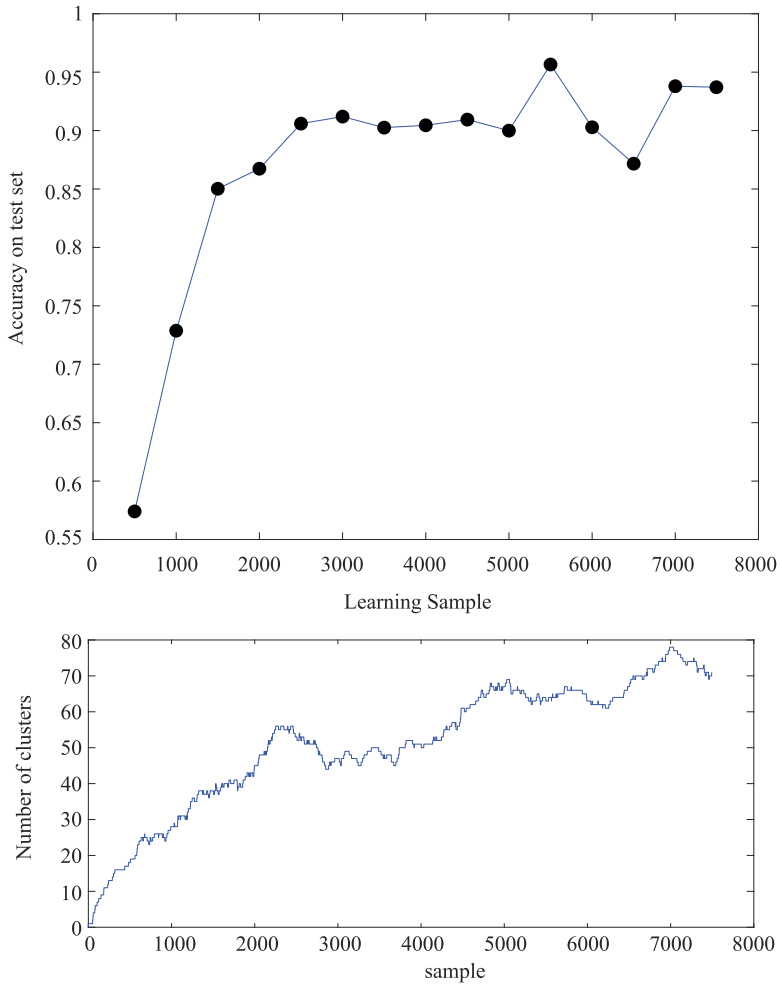

Fig. 18. Classification accuracy of digits test set depending on the number of processed samples of learning set and cluster evolution using Mahalanobis distance.

Table 12
Results for egg classification.

Method	Clusters	Accuracy
FLEXFIS-Cl. SM	NA	0.8801
FLEXFIS-Cl. SM with LOFO	NA	0.9011
FLEXFIS-Cl. SM with feat.-wise	NA	0.9155
FLEXFIS-Cl. MM	NA	0.9412
FLEXFIS-Cl. MM with LOFO	NA	0.9655
FLEXFIS-Cl. MM with feat.-wise	NA	0.9722
eC_e	22	0.9666
eC_m	13	0.9674

with a partial covariance matrix and partial eigenvectors matrix. In most cases, the density measure based on the Euclidean distance produced slightly worse results than the Mahalanobis norm. However, the Euclidean norm is more robust and easier to tune. The problem with the Mahalanobis norm is in the initialization of the covariance matrix (factor α), which strongly influences the results. Furthermore, the Mahalanobis norm is more sensitive to linear dependency among the input variables. The presented eCauchy methodology is based on non-normalized values of densities and employs a winner takes all (if it is close enough) policy. This makes the method slightly less accurate (especially in regression problems) in comparison to methods that employ normalization (e.g. eFuMo), but in contrast, the method is less affected by noisy data and outliers. From the mathematical point of view, the hyperplane procedure for the identification of local models is a very elegant solution. However, in practice, there are some problems that affect the performance of the model. The method usually works fine for small dimensional static problems for which the inputs are carefully chosen to avoid linear dependency

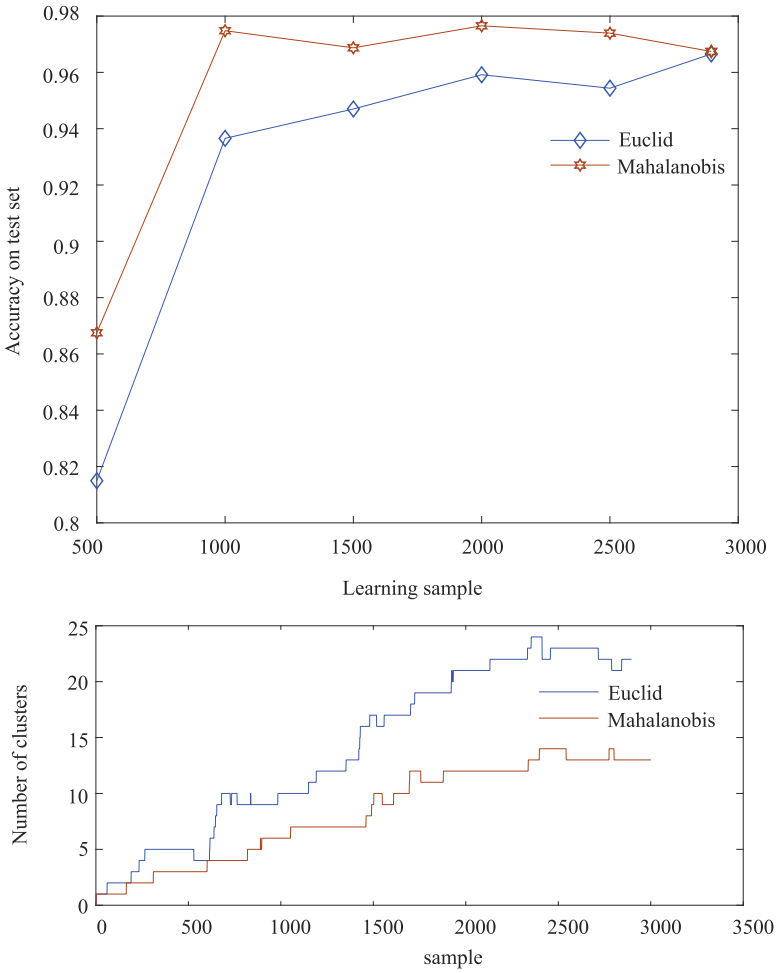

Fig. 19. Classification accuracy of test set depending on the number of processed samples of learning set and cluster evolution for egg classification example.

among them. The method is also susceptible to the scales of the variables. To achieve good results the variables should have the same range. In this sense, the recursive least squares method is more robust and more suitable for practical applications (Fig. 19).

There are several high-dimensional data streams tested with the results reported in Section 5, where it turned out that our method achieved comparable or even better performance than several SoA (i.e. State-of-the-Art) methods – for the purpose of clarity an overview table over the data streams and their characteristics is given in Table 13. The overview of results over different streaming datasets is given regarding the number of features, number of used samples, number of generated classes (and output errors in the case of regression), noise level, type of data stream, and the average calculation time over the number of clusters.

In Table 14 the performance of the proposed method is evaluated. The accuracy improvement over the best SoA method subject to respecting the eventual complexity increase is given for different datasets. For example, for Box–Jenkins gas furnace results, the best SoA method regarding accuracy is DENFIS with 0.0190, thus the performance improvement regarding our method is -35% (thus, an accuracy decrease), however the complexity is reduced from 12 to 3 clusters by the proposed method thus by 75%, hence the entry is -35/75. At the end, the average improvement over the best SoA method is given for each dataset. It is shown that the proposed method gives slightly worse accuracy with lower complexity of the model.

Future work will focus on the inverse of the partial covariance matrix and partial eigenvectors matrix norms. In this paper only latent vector was used to form a norm. This did not work well for higher dimensional problems. However, including more eigenvectors might improve results for higher dimensions and also form a mechanism for feature extraction.

Table 13
An overview table of results obtained by eCauchy method over different data streams and their characteristics.

Data stream	Features	Samples	Classes	Noise level	Comment	Average time
Classification			Number			
Iris data	4	150–4050	3	Low	Real benchmark	0.3 ms
Iris data	4	150–4050	3	Low added	Real benchmark	0.3 ms
Iris data	4	150–4050	3	Medium added	Real benchmark	0.4 ms
Iris data	4	150–4050	3	High added	Real benchmark	0.5 ms
Pen-based recognition	16	7494	10	Medium	Real data	8 ms
Eggs classification	19	2895	2	Low	Real data	2 ms
Laser data	2	180	5	Low	Real data	0.2 ms
Clustering example	2	80	5	Low	Artificial data	0.2 ms
Regression			Error (min-max, std)			
Mackey–Glass time series	5	3000	0.42 - 1.31, 0.22	None	Simulated benchmark	0.6 ms
Mackey–Glass time series	5	3000	−1.11 - 3.29, 0.26	Medium added	Simulated benchmark	0.6 ms
Function example I	2	200	−5.79 - 10.18, 1.73	High	Simulated benchmark	0.2 ms
Function example II	2	10,000	−4.17 - 4.04, 0.94	Low added	Simulated benchmark	1.2 ms
Function example II	2	10,000	−4.50 - 4.25, 1.24	Medium added	Simulated benchmark	1.2 ms
Box–Jenkins example	4	50,000	45.60 - 60.50, 3.20	None	Simulated benchmark	1.2 ms
Rolling mill	7	13,155	50.35 - 311.97, 74.21	Medium	Real data	2 ms

Table 14
Comparison between the best SoA methods and the proposed method.

Dataset	Dimensionality of learning problem	Accuracy/complexity comparison with best SoA method (in %)
Box–Jenkins example2	4	−2.5/25
Box–Jenkins example1	4	−35/75
Rolling-Mill	7	−19/89
Mackey–Glass time series	5	−11/47
Eggs classification	19	−0.5/NA
Pen-based recognition	16	−2.5/−16
Average improvement		**−15.5/38**

7. Conclusion

The main contribution of the proposed paper is to present the method which unify and generalize the evolving clustering based on Cauchy density. The proposed approach can deal with different classification or regression problems easily, just by using suitable inner matrix norm in the density definition. The used normalized Cauchy density enables very easy tuning of important thresholds of the method, Γ_{min} and Γ_{max}, which are then very similar for a wide range of different problems and therefore can be easily tuned. The proposed method represents a very successful framework for a wide range of problems and can easily cope with problems with outliers, noise, clusters of different volumes and different shapes. The evolving nature makes the algorithm suitable for big-data problems. The basic features of the proposed algorithm were presented on simple school examples, selected benchmark problems, and on real data streams. The obtained results are collected in Table 14 where the comparison between the proposed method and the main state-of-the-art methods is given. The comparison is given for different data streams and different methods. For each data stream the best method is detected according to the accuracy and complexity and compared to the proposed method. It is shown that in average our method follows the best solutions (i.e. in average just 15% behind the best method for each data stream) with much lower complexity. The proposed method offers framework for classification and regression problems.

Acknowledgements

This work has been supported by Slovenian Research Agency with the research programme P2-0219, Modelling, simulation and control and the Austrian COMET-K2 (K24301) program of the Linz Center of Mechatronics (LCM), funded by the Austrian federal government and the federal state of Upper Austria, and of the Linz Austrian research funding association (FFG) within the scope of the programme IKT of the future, the project of Generating process feedback from heterogeneous data sources in quality control.

Appendix A

Further developing of the summation from Eq. (2) is performed by adding and subtracting the column vector of cluster mean value $\boldsymbol{\mu}^j$ to the factors on both sides of the inner matrix norm $\boldsymbol{A}^j$:

$$\sum_{i=1}^{M^j}(\boldsymbol{z}(k)-\boldsymbol{z}_i^j)^T\boldsymbol{A}^j(\boldsymbol{z}(k)-\boldsymbol{z}_i^j)=M^j(\boldsymbol{z}(k)-\boldsymbol{\mu}^j)^T\boldsymbol{A}^j(\boldsymbol{z}(k)-\boldsymbol{\mu}^j)$$

$$-2\sum_{i=1}^{M^j}(\boldsymbol{z}_i^j-\boldsymbol{\mu}^j)^T\boldsymbol{A}^j(\boldsymbol{z}(k)-\boldsymbol{\mu}^j)$$
$$+\sum_{i=1}^{M^j}(\boldsymbol{z}_i^j-\boldsymbol{\mu}^j)^T\boldsymbol{A}^j(\boldsymbol{z}_i^j-\boldsymbol{\mu}^j) \tag{33}$$

where

$$\boldsymbol{\mu}^j=\frac{1}{M^j}\sum_{i=1}^{M^j}\boldsymbol{z}_i^j \tag{34}$$

is defined as the center of the jth cluster and where lower and upper index at $\boldsymbol{z}_i^j$ define ith sample in jth cluster. It is easy to show that the last term of Eq. (33) is identical to

$$\sum_{i=1}^{M^j}(\boldsymbol{z}_i^j-\boldsymbol{\mu}^j)^T\boldsymbol{A}^j(\boldsymbol{z}_i^j-\boldsymbol{\mu}^j)=(M^j-1)\,\mathrm{trace}(\boldsymbol{A}^j\boldsymbol{\Sigma}^j) \tag{35}$$

where

$$\boldsymbol{\Sigma}^j=\frac{1}{M^j-1}\sum_{i=1}^{M^j}(\boldsymbol{z}_i^j-\boldsymbol{\mu}^j)(\boldsymbol{z}_i^j-\boldsymbol{\mu}^j)^T \tag{36}$$

represents the covariance matrix of jth cluster which has M^j samples. The second term in Eq. (33) is identical to 0 due to the definition of $\boldsymbol{\mu}^j$ in Eq. (34). The notation of the covariance matrix of the jth cluster can be also written as $\boldsymbol{\Sigma}_{M^j}^j$ to explicitly express that the covariance was calculated for M^j samples. The weighted Cauchy density in Eq. (2) therefore takes the form which is suitable for recursive implementation (similar as in [9]).

Appendix B

To calculate different densities proposed in Section 2, the cluster covariance matrix is needed. The definition of covariance matrix (36) is not suitable for implementation in the recursive identification algorithm because all past cluster data are needed. For an on-line identification algorithm, the covariance matrix must be calculated recursively. In the presented approach a sample is either fully assigned to a certain cluster or it is not assigned at all (and treated as an outlier or a new cluster depending on a problem). When a new data sample, denoted as $\boldsymbol{z}(k)$, is assigned to the jth cluster, the number of samples is incremented, and the mean and the covariance matrix are updated [9]. In order to make the notation clear, cluster center will be denoted as $\boldsymbol{\mu}_{M^j}^j$ where needed, to specify that the jth cluster consists of M^j samples.

First, the difference between the current sample and the current mean value is calculated as:

$$\boldsymbol{e}_{M^j}^j=\boldsymbol{z}(k)-\boldsymbol{\mu}_{M^j}^j \tag{37}$$

Next, the mean is updated

$$\boldsymbol{\mu}_{M^j+1}^j=\boldsymbol{\mu}_{M^j}^j+\frac{1}{M^j+1}\boldsymbol{e}_{M^j}^j \tag{38}$$

After that, the un-normalized covariance matrix is computed:

$$\boldsymbol{S}_{M^j+1}^j=\boldsymbol{S}_{M^j}^j+\boldsymbol{e}_{M^j}^j(\boldsymbol{z}(k)-\boldsymbol{\mu}_{M^j+1}^j)^T \tag{39}$$

The covariance matrix is then obtained as:

$$\boldsymbol{\Sigma}_{M^j+1}^j=\frac{1}{M^j}\boldsymbol{S}_{M^j+1}^j \tag{40}$$

It should be mentioned that the covariances of the clusters are calculated with the winner-takes-all approach. The states ($\boldsymbol{\mu}^j$ and $\boldsymbol{S}^j$) of this algorithm are initialized with zeros, which can lead to the problem of invertibility of $\boldsymbol{\Sigma}^j$ in the early phase. To avoid this, $\boldsymbol{S}^j$ can be initially set to the identity matrix multiplied by a small positive number.

References

[1] J. Abonyi, B. Feill, Cluster Analysis for Data Mining and System Identification, Birkhuser Basel, 2007.
[2] P. Angelov, D.P. Filev, An approach to online identification of Takagi–Sugeno fuzzy models, IEEE Trans. Syst. Man Cyber. Part B 34 (1) (2004) 484–497.
[3] P. Angelov, P. Angelov, D. Filev, A. Kasabov, Evolving Takagi–Sugeno fuzzy systems from streaming data (ets+), in: Evolving Intelligent Systems: Methodology and Applications, in: IEEE Press Series on Computational Intelligence, John Willey and Sons, 2010, pp. 273–300.
[4] P. Angelov, R. Yager, Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density, in: 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS), 2011, pp. 62–69.
[5] P. Angelov, R. Yager, A new type of simplified fuzzy rule-based system, Int. J. Gen. Syst. 41 (2) (2011) 163–185.
[6] P. Angelov, X. Gu, G. Gutierrez, J.A. Iglesias, A. Sanchis, Autonomous data density based clustering method, in: The bi-annual IEEE World Congress on Computational Intelligence (IEEE WCCI), 2016, pp. 1–9.

[7] R. Babuška, P.J. van der Veen, U. Kaymak, Improved covariance estimation for Gustafson-Kessel clustering, in: FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, 2002, Honolulu, HI, 2002, pp. 1081–1085.
[8] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.
[9] S. Blažič, D. Dovzăn, I. Skrjanc, Cloud-based identification of an evolving system with supervisory mechanisms, in: IEEE Control Systems Society Multiconference on Systems and Control, Nice/Antibes, France, 2014, pp. 1906–1911.
[10] D. Dovžan, I. Škrjanc, Recursive fuzzy c-means clustering for recursive fuzzy identification of time-varying processes, ISA Trans. 50 (2) (2011) 159–169.
[11] D. Dovžan, I. Škrjanc, Recursive clustering based on a Gustafson-Kessel algorithm, Evol. Syst. 2 (2011) 15–24.
[12] D. Dovžan, V. Logar, I.v. Skrjanc, Implementation of an evolving fuzzy model (efumo) in a monitoring system for a waste-water treatment process, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1761–1776.
[13] J.C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well separated cluster, J. Cybern. 3 (1974) 32–57.
[14] D. Filev, O. Georgieva, An extended version of the Gustafson-Kessel algorithm for evolving data stream clustering, in: P. Angelov, D. Filev, A. Kasabov (Eds.), Evolving Intelligent Systems: Methodology and Applications, IEEE Press Series on Computational Intellegence, John Willey and Sons, 2010, pp. 273–300.
[15] D.E. Gustafson, W.C. Kessel, Fuzzy clustering with a fuzzy covariance matrix, in: IEEE CDC, San Diego, CA, USA, 1979, pp. 761–766.
[16] F. Hoppner, F. Klawon, Improved fuzzy partitions for fuzzy regression models, J. Approximate Reasoning 32 (2003) 85–102.
[17] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: IEEE International Joint Conference on Neural Networks, 2004, pp. 985–990.
[18] C.F. Juang, C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Syst. 6 (1) (1998) 12–32.
[19] V. Kadirkamanathan, M. Niranjan, A function estimation approach to sequential learning with neural networks, Neural Comput. 5 (6) (1993) 954–975.
[20] N. Kasabov, Evolving fuzzy neural networks for supervised/unsupervised on-line knowledge-based learning, IEEE Trans. Syst. Man Cyber. Part B 31 (6) (2001) 902–918.
[21] N. Kasabov, Q. Song, DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Trans. Fuzzy Syst. 10 (2) (2002) 144–154.
[22] U. Kaymak, M. Setnes, Fuzzy clustering with volume prototypes and adaptive cluster merging, IEEE Trans. Fuzzy Syst. 10 (6) (2002) 705–712.
[23] G. Klančar, I.v. Skrjanc, Evolving principal component clustering with a low run-time complexity for LRF data mapping, Appl. Soft Comput. 35 (2015) 349–358.
[24] R. Krishnapura, J.M. Keller, Possibilistic approach to clustering, IEEE Trans. Fuzzy Syst. 1 (1993) 98–100.
[25] G. Leng, G. Prasad, T.M. McGinnity, An on-line algorithm for creating self-organizing fuzzy neural networks, Neural Netw. 17 (2004) 1477–1493.
[26] E. Lima, M. Hell, R. Ballini, F. Gomide, Evolving fuzzy modeling using participatory learning, Evolving intelligent systems: Methodology and Applications, 2010, Chapter 4, doi:10.1002/9780470569962.ch4.
[27] E. Lughofer, P. Angelov, X. Zhou, Evolving single- and multi-model fuzzy classifiers with FLEXFIS-class, in: 2007 IEEE International Fuzzy Systems Conference, London, 2007, pp. 1–6.
[28] E. Lughofer, 1–23., On-line incremental feature weighting in evolving fuzzy classifiers, Fuzzy Sets Syst. 163 (2011).
[29] E. Lughofer, Dynamic evolving cluster models using on-line split-and-merge operations, in: 10th IEEE International Conference on Machine Learning and Applications, 2011, pp. 20–26.
[30] E. Lughofer, A dynamic split-and-merge approach for evolving cluster models, Evol. Syst. 3 (3) (2012) 135–151.
[31] E. Lughofer, C. Cernuda, S. Kindermann, M. Pratama, Generalized smart evolving fuzzy systems, Evol. Syst. (6) (2015) 269–292.
[32] E. Lughofer, M. Pratama, I.v. Škrjanc, Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation, IEEE Transactions on Fuzzy Systems, 2018.
[33] L. Maciel, F. Gomide, R. Ballini, Recursive possibilistic fuzzy modeling, in: Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on, IEEE SSCI, Orlando, 2014, pp. 9–16.
[34] B. Ojeda-Magana, R. Ruelas, M.A. Corona-Nakamura, D. Andina, An improvement to the possibilistic fuzzy c-means clustering algorithm, Intell. Autom. Soft Comput. 20 (1) (2006) 585–592.
[35] N.R. Pal, K. Pal, J.M. Keller, J.C. Bezdek, A possibilistic fuzzy c-means clustering algorithm, IEEE Trans. Fuzzy Syst. 13 (4) (2005) 517–530.
[36] M. Pratama, S. Anavatti, P. Angelov, E. Lughofer, PANFIS: A novel incremental learning machine, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 55–68.
[37] J.A. Ramey, P.D. Young, A comparison of regularization methods applied to the linear discriminant function with high-dimensional microarray data, J. Stat. Comput. Simul. 83 (3) (2013) 581–596.
[38] H.J. Rong, N. Sundararajan, G.B. Huang, P. Saratchandran, Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction, Fuzzy Sets Syst. 157 (9) (2006) 1260–1275.
[39] R. Rosa, R. Ballini, F. Gomide, Evolving hybrid neural fuzzy network for system modeling and time series forecasting, in: IEEE International Conference on Machine Learning and Applications, Miami, 2013, pp. 4–7.
[40] R. Rosa, F. Gomide, D. Dovžan, I. Škrjanc, Evolving neural network with extreme learning for system modeling, in: Proceedings of The 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems, Linz, Austria, 2-4 June, 2014, pp. 11–34.
[41] J. Rubio, SOFMLS: on-line self-organizing fuzzy modified least-squares network, IEEE Trans. Fuzzy Syst. 17 (6) (2009) 1296–1309.
[42] S.-B. H., C. Lucas, B.N. Araabi, Recursive Gath-Geva clustering as a basis for evolving neuro-fuzzy modeling, in: Evolving Systems, volume 1, Springer, 2010, pp. 59–71.
[43] I. Škrjanc, Evolving fuzzy-model-based design of experiments with supervised hierarchical clustering, IEEE Trans. Fuzzy Syst. 23 (4) (2015) 861–871.
[44] I. Škrjanc, D.D. zan, Evolving Gustafson-Kessel possibilistic c-means clustering, INNS Conf. Big Data 53 (2015) 191–198.
[45] H. Timm, C. Borgelt, C. Doering, R. Kruse, An extension to possibilistic fuzzy cluster analysis, Fuzzy Sets Syst. 147 (1) (2004) 3–16.
[46] S.W. Tung, C. Quek, C. Guan, SaFIN: a self-adaptive fuzzy inference network, IEEE Trans. Neural Netw. 22 (12) (2011) 1928–1940.
[47] L. Yingwei, N. Sundararajan, P. Saratchandran, A sequential learning scheme for function approximation using minimal radial basis function (RBF) neural networks, Neural Comput. 9 (1997) 461–478.