Evolving Clustering, Classification and Regression with TEDA

Dmitry Kangin*, Plamen Angelov*,**
* Data Science Group, School of Computing and Communications, Lancaster University, UK
** Chair of Excellence, University Carlos III, Madrid, Spain

{d.kangin, p.angelov}@lancaster.ac.uk

***Abstract*— In this article the novel clustering and regression methods TEDACluster and TEDAPredict methods are described additionally to recently proposed evolving classifier TEDAClass. The algorithms for classification, clustering and regression are based on the recently proposed AnYa type fuzzy rule based system. The novel methods use the recently proposed TEDA framework capable of recursive processing of large amounts of data. The framework is capable of computationally cheap exact update of data per sample, and can be used for training 'from scratch'. All three algorithms are evolving that is they are capable of changing its own structure during the update stage, which allows to follow the changes within the model pattern.**

Keywords—evolving systems; clustering; classification; regression; incremental update.

I. INTRODUCTION

Nowadays, there are many general or specialised algorithms for different machine learning problems: clustering, classification, regression, anomaly detection. Some of them are aimed for one particular problem, other are specified for a wide scope of problem formulations. These algorithms employ different frameworks (Support Vector Machines (SVM) [1], neural networks [2], fuzzy systems [3], decision trees [4] for classification, Support Vector Regression (SVR) [5], [6], mixture models [7], fuzzy systems [8] for regression, mixture models [7], MacQueen's *k*-means [9], and fuzzy systems [10] for clustering).

However, despite the architectural differences and the statement, they can be grouped with respect to the online processing functionality they offer. Some of the algorithms are *offline*. It means that any change within the data set used for algorithm adjustment will necessarily need full re-training.

Other algorithms, referred as *'incremental'*, offer the opportunity to train the system only with the new data, given the algorithm trained for all the previous data. It should be emphasised that there are no requirements on how the previous data can be utilised in the training process. The update state may be either exact or approximate, depending on the given classifier. Therefore, the algorithm can hold as much data from the previous iterations as it is needed.

Some incremental algorithms additionally do not use all the previous data samples, but some context of the system state, having fixed memory size upper bound. These algorithms are referred as *'online'*. The main advantage of such algorithms is the independence of the memory consumption requirements of the overall training set size, which is especially useful for data stream processing.

Finally, there is a group of algorithms, which is referred as *'evolving'* [11], which also hold only fixed-size context of the system, but it is capable of discarding previous knowledge additionally to adding new data, i.e. changing the structure of the system by addition or deletion of data structures describing the system. The evolving systems are viewed basically in the context of data streams, as it is especially useful to address 'shift' and 'drift' concepts [11], [12], describing the changes of statistical characteristics of the data stream in time. Basically, the term has arisen in application to the fuzzy systems frameworks [13], but it can be applied to neural networks, particularly evolving spiking neural networks [14], SVM models [15] and other systems. Hence apart of the previous algorithms, the evolving algorithms allow to follow the changes within the pattern by the model adaptation.

The family of fuzzy systems proposed in this article belongs to the last group of the algorithms. It is evolving, as it is capable of adding and removing the clusters during the data stream processing that is significantly useful for various applications. Basically, the fuzzy systems follow the same paradigm as previously proposed *eClass* [16] and *AutoClass* [17] frameworks.

The next sections are given as follows. The second section gives a review of the fuzzy systems. The third section recently proposed statistical data processing framework TEDA (acronym from *Typicality and Eccentricity Data Analysis*). The fourth section describes the details for TEDACluster algorithm for clustering. The fifth section describes the TEDAClass algorithm for classification, previously proposed in [18]. The sixth section presents the TEDAPredict algorithm for regression. Then, the experimental section is given, revealing the results given for the proposed algorithm. Finally, the conclusion is given.

II. FUZZY SYSTEMS REVIEW

The fuzzy sets theory and fuzzy rule based systems were developed during last half-century. The theory arose from the seminal work of Lotfi Zadeh in 1965 [19]. Several major types

978-1-4799-1959-8/15/$31.00

of fuzzy system were proposed. Zadeh- Mamdani fuzzy system was proposed by L. Zadeh [20] and E. Mamdani [21]. Takagi-Sugeno fuzzy system was proposed by T.Takagi and Sugeno in [22].
These fuzzy rule systems are formulated as follows:
Mamdani:

$$IF\ (ant_i(\boldsymbol{x}))\ THEN\ (y_i\ IS\ Y_i), \tag{1}$$

Takagi-Sugeno:

$$IF\ (ant_i(\boldsymbol{x}))\ THEN\ (y_i = \boldsymbol{x}^T\Theta_i). \tag{2}$$

where y_i is an outcome of the i-th rule, $i \in [1..N]$, Θ_i is a design matrix for linear regression, $\boldsymbol{x}$ is an input data vector, $\text{ant}_i(\boldsymbol{x})$ is the antecedent of the fuzzy rule.

The antecedent part of the rules in both the systems is the same, and it is expressed as

$$\text{ant}_i(\boldsymbol{x})\text{: } x^1 is\ L_i^1\ AND\ x^2 is\ L_i^2\ AND\ \dots AND\ x^n is\ L_n^i, \tag{3}$$

where n is a dimensionality of the vector $\boldsymbol{x} = (x^1, x^2, \dots x^n)$,
$\boldsymbol{L}^i = (L_1^i, L_2^i \dots L_n^i)$ is a reference vector for the fuzzy rule.
Hence, the antecedent is combined of the fuzzy sets or membership functions for each of the vector components.
Therefore, instead of these fuzzy rule systems, we use AnYa [23] fuzzy rule system, which will be described after, as it gives more general expressions for multi-dimensional variables.
In AnYa fuzzy rule system, each of the rules is formulated as

$$\text{AnYa: } IF\ (\boldsymbol{x}\ \text{is like}\ \boldsymbol{L}_i)\ THEN\ (y_i = \boldsymbol{x}^T\Theta_i). \tag{4}$$

Here 'like' is some predicate, depending of the problem formulation, which gives more flexibility to the machine learning algorithms we build up.

Nowadays, a lot of fuzzy systems are serving to solve classification, regression and clustering problems. Recently introduced algorithms include DEC [24], AutoClass [17], eClass [16], FLEXFISClass [25]. Some of these algorithms are included in comparison in the experimental section.

The algorithm we propose here is different in terms that it relies on the recently proposed TEDA statistical framework.

III. TEDA DESCRIPTION

After giving a brief introduction into fuzzy rule systems, we describe the foundations of the algorithms we use to build an antecedent for the fuzzy rule. To do this, we discuss a recently proposed framework TEDA for a 'per point' online data analysis.
We start from defining the feature space $\mathfrak{X} \subseteq \mathbb{R}^n$ which contains the data samples. We define some distance $d(\boldsymbol{x}, \boldsymbol{y})$ within the space. It can be set according to any metric or, broader, similarity, i.e. Manhattan (L_1), Euclidean, Mahalanobis [26], cosine.

Consider the (theoretically infinite, practically as large as it is needed) data samples sequence

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2 \dots \boldsymbol{x}_k \dots\}, \boldsymbol{x}_k \in \mathfrak{X}, k \in \mathbb{N}. \tag{5}$$

Here k is an index within the sequence, which can be interpreted as a time stamp for the data sample arrival time.

For all the data stream we define the sum distance for the points of the sequence up to the k-th sequence element:

$$\pi^k(\boldsymbol{x}) = \sum_{i=1}^{k} d(\boldsymbol{x}, \boldsymbol{x}_i). \tag{6}$$

It is easy to see that

$$\pi^k(\boldsymbol{x}) = \pi^{k-1}(\boldsymbol{x}) + d(\boldsymbol{x}, \boldsymbol{x}_i), k \geq 2. \tag{7}$$

Then, we define two characteristics of the data sequence, complementary to one another.

First of them, eccentricity, is defined as

$$\xi^k(\boldsymbol{x}) = \frac{2\pi^k(\boldsymbol{x})}{\sum_{i=1}^{k}\pi^k(\boldsymbol{x}_i)} = 2\frac{\sum_{i=1}^{k} d(\boldsymbol{x}, \boldsymbol{x}_j)}{\sum_{i=1}^{k}\sum_{j=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)}, \tag{8}$$

$$k \geq 2,$$

$$\sum_{i=1}^{k}\pi^k(\boldsymbol{x}) > 0.$$

Eccentricity is bounded:

$$0 \leq \xi^k(\boldsymbol{x}) \leq 1, \sum_{i=1}^{k}\xi^k(\boldsymbol{x}_i) = 2. \tag{9}$$

Second characteristic, typicality, is defined as

$$\tau^k(\boldsymbol{x}) = 1 - \xi^k(\boldsymbol{x}). \tag{10}$$

Similarly to eccentricity, typicality is bounded:

$$0 \leq \tau^k(\boldsymbol{x}) \leq 1, \sum_{i=1}^{k}\tau^k(\boldsymbol{x}_i) = k - 2, \tag{11}$$

$$k \geq 2.$$

Hence, eccentricity and typicality can have also normalised versions:

$$\zeta^k(\boldsymbol{x}) = \frac{\xi^k(\boldsymbol{x})}{2}, \sum_{i=1}^{k} \zeta^k(\boldsymbol{x}_i) = 1, k \geq 2, \quad (12)$$

$$t^k(\boldsymbol{x}) = \frac{\tau^k(\boldsymbol{x})}{k-2}, \sum_{i=1}^{k} \tau^k(\boldsymbol{x}_i) = 1, k > 2.$$

For any distance, the typicality and eccentricity can be calculated recursively using formula (7) [18]. For Euclidean and Mahalanobis distances, it can be calculated recursively by another special way, which avoids computational issues with this method caused by the growth of the accumulated distance.

For Eucludean distance,

$$\xi^k(\boldsymbol{x}) = 2\frac{\sum_{i=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{i=1}^{k}\sum_{j=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)} = \quad (13)$$

$$= 2\frac{\sum_{i=1}^{k}(\boldsymbol{x}_i - \boldsymbol{x})^T(\boldsymbol{x}_i - \boldsymbol{x})}{\sum_{i=1}^{k}\sum_{j=1}^{k}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j)}$$

can be replaced by [18]

$$\xi^k(\boldsymbol{x}) = \frac{1}{k} + \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k[\sigma_x^k]^2}. \quad (14)$$

Mean $\boldsymbol{\mu}_x^k$ and variance σ_x^k can be calculated recursively using the formulae:

$$\boldsymbol{\mu}_x^k = \frac{(k-1)\boldsymbol{\mu}_x^{k-1}}{k} + \frac{\boldsymbol{x}_k}{k}, k \geq 1, \boldsymbol{\mu}_x^0 = \mathbf{0}. \quad (15)$$

$$\mu_{x^T x}^k = \frac{(k-1)\mu_{x^T x}^{k-1}}{k} + \frac{\boldsymbol{x}_k^T\boldsymbol{x}_k}{k}, k \geq 1, \mu_{x^T x}^0 = 0, \quad (16)$$

$$[\sigma_x^k]^2 = \mu_{x^T x}^k - [\boldsymbol{\mu}_x^k]^T\boldsymbol{\mu}_x.$$

The typicality, similarly, can be represented as

$$\tau^k(x) = 1 - \xi^k(x) = \frac{k-1}{k} - \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k[\sigma_x^k]^2}. \quad (17)$$

For the Mahalanobis case [18],

$$\xi^k(\boldsymbol{x}) = 2\frac{\sum_{i=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{i=1}^{k}\sum_{j=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)} = \quad (18)$$

$$= 2\frac{\sum_{i=1}^{k}(\boldsymbol{x} - \boldsymbol{x}_i)^T\left[\Sigma_x^k\right]^{-1}(\boldsymbol{x} - \boldsymbol{x}_i)}{\sum_{i=1}^{k}\sum_{j=1}^{k}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T\left[\Sigma_x^k\right]^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j)},$$

where covariance matrix is defined as

$$[\Sigma_x^k] = \frac{1}{k}\sum_{m=1}^{k}(\boldsymbol{x}_m - \boldsymbol{\mu}_x^k)(\boldsymbol{x}_m - \boldsymbol{\mu}_x^k)^T. \quad (19)$$

The covariance matrix can be updated recursively using Woodbury formula, for derivation see [18].

$$\left[\Sigma_x^k\right]^{-1} = p^{-1} + \frac{p^{-1}\frac{(k-1)\boldsymbol{\mu}_x^{k-1}}{k^2}\left(\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k\right)^T p^{-1}}{1 + \frac{(k-1)}{k^2}\left(\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k\right)^T p^{-1}\boldsymbol{\mu}_x^{k-1}}, \text{ where} \quad (20)$$

$$p^{-1} = \quad (21)$$

$$= \frac{\left(\frac{k}{k-1}\right)[\Sigma_x^{k-1}]^{-1} + \frac{k+1}{k-1}[\Sigma_x^{k-1}]^{-1}x_k x_k^T[\Sigma_x^{k-1}]^{-1}}{1 + \frac{k+1}{k}x_k^T[\Sigma_x^{k-1}]^{-1}x_k}.$$

Then it can be proven [18] that

$$\xi^k(\boldsymbol{x}) = \frac{(\boldsymbol{x} - \boldsymbol{\mu}_x^k)^T\left[\Sigma_x^k\right]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x^k)}{kn} + \frac{1}{k}. \quad (22)$$

The corresponding equation for typicality is

$$\tau^k(\boldsymbol{x}) = \frac{k-1}{k} - \frac{(\boldsymbol{x} - \boldsymbol{\mu}_x^k)^T\left[\Sigma_x^k\right]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x^k)}{kn}. \quad (23)$$

It can be proven also, that the restrictions on typicality are equivalent to the Chebyshev inequality:

$$t^k(x) = \frac{1}{k-1} - \frac{(\boldsymbol{x} - \boldsymbol{\mu}_x^k)^T\left[\Sigma_x^k\right]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x^k)}{kn(k-1)} - \quad (24)$$

$$-\frac{1}{kn(k-1)} > T(k) = \frac{1}{k-1} - \frac{m^2 + n}{kn(k-1)},$$

where $T(k)$ is a threshold, m is some variable, which gives the area of tolerance, is equivalent to

$$(\boldsymbol{x} - \boldsymbol{\mu}_x^k)^T[\Sigma_x^k]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x^k) < m^2. \quad (25)$$

IV. TEDACLUSTER

In this section, we propose *TEDACluster* clustering algorithm. Actually, it is based on the cloud concept, which is a key concept for the *AnYa* fuzzy rule system. According to the concept, each of the elements of data sequence is described in terms of fuzzy membership which is given by like predicate $\boldsymbol{x}\sim\boldsymbol{x}_i^*$. Like a cluster, it is a group of similar data points, but it is not bounded or shaped specifically.

Consider fuzzy rule system:

$$F = \{R_i\}, i = \overline{1, N},$$
$$R_i(\boldsymbol{x}): IF\ (\boldsymbol{x}\sim\boldsymbol{x}_i^*)\ THEN\ i, \quad (26)$$

where F is a fuzzy rule system, consisting of the rules R_i, i is the index of the rule, $\boldsymbol{x} \in \mathfrak{X}, F: \mathfrak{X} \to K$, $\mathfrak{X}$ is a feature (object) space, K is a cluster set, $\boldsymbol{x}_i^*$ is a representative point (focal point) for the cluster, $\sim$ is a *like* predicate, which reflects some relation of 'closeness' (association/membership) of the point $\boldsymbol{x}$ to the focal point of the fuzzy rule $R_i(\boldsymbol{x})$.

Then, to compose the evolving algorithm, we need to define the *like* predicate, as well as some rules for addition and deletion of the data within the framework.

First, we begin from the *like* predicate definition. We base it on the same propositions as it is done in [18]. We can notice that the larger is the typicality value, the closer the object is to the cloud. We define the firing strength of each rule is determined as

$$w_i^k(\boldsymbol{x}) = \frac{t_i^k(\boldsymbol{x})}{\sum_{j=1}^{N} t_j^k(\boldsymbol{x})}, \tag{27}$$

where normalised typicality t_i^k is given over all the objects assigned to the fuzzy rule. The new object is assigned to the fuzzy rule for which the firing strength is defined, and then the typicality is recalculated according to the formula [18].

Then, we define the procedure for addition and deletion of rules within the fuzzy rules system.

As it was stated in the previous work on *TEDAClass* [18], and contrary to the formulations of *eClass* [16] and *AutoClass* [17], we do not define any global statistical characteristics over the data.

We create a new data cloud based on local typicality. We create a new rule, when the typicality for all the clusters is less that the given threshold, given w.r.t. the equation (24):

$$\forall\, R^i \in F\ t_i^k(\boldsymbol{x}) < T(k) \Leftrightarrow \mathfrak{Y}(k) = 1. \tag{28}$$

$\mathfrak{Y}(k) \in \{0,1\}$ shows whether the cluster should be created. Also, all the rules which are closer than $T(k)$ to the newly created rule, should be deleted:

$$R^i \in F: t_i^k(\boldsymbol{x}_k) > T(k). \tag{29}$$

The deletion includes merging the parameters for each of the clusters.

The merging of clusters is made according to the formulae, which can be proven as an extension of the ordinary cluster update:

$$\boldsymbol{\mu}_l^k = \frac{\mathrm{Support}(R_\mathrm{i})\boldsymbol{\mu}_i^k}{\mathrm{Support}(R_\mathrm{i}) + \mathrm{Support}(R_\mathrm{j})} + \frac{\mathrm{Support}(R_\mathrm{j})\boldsymbol{\mu}_j^k}{\mathrm{Support}(R_\mathrm{i}) + \mathrm{Support}(R_\mathrm{j})}, k \geq 1. \tag{30}$$

Here, the function $\mathrm{Support}(R_\mathrm{i})$ gives the support of the cluster, given as a number of objects assigned for mean update for this cluster.

The variance update is performed using the formula

$$\boldsymbol{\mu}_{x^T x,l}^k = \frac{\mathrm{Support}(R_\mathrm{i})\boldsymbol{\mu}_{x^T x,i}^k}{\mathrm{Support}(R_\mathrm{i}) + \mathrm{Support}(R_\mathrm{j})} + \frac{\mathrm{Support}(R_\mathrm{j})\boldsymbol{\mu}_{x^T x,j}^k}{\mathrm{Support}(R_\mathrm{i}) + \mathrm{Support}(R_\mathrm{j})}, k \geq 1. \tag{31}$$

The covariance update is based on Woodbury formula and can be done using Woodbury formula [18].

The final clustering algorithm is given in Figure 1.

k = 1; $K = \emptyset$
While the algorithm is not terminated do
Wait for the new sample x_k;
Cluster_ID = Execute "Find Cluster" (x_k, K);
If (Cluster_ID is $\emptyset$)
Cluster_ID = Execute "Create Cluster" (x_k, K);
End
Execute "Update Clusters"$(x_k, K_{\mathrm{Cluster\_ID}})$;
k = k + 1.
End
Function Cluster_ID = "Find Cluster" $(\boldsymbol{x}_k, K)$
If (K is $\emptyset$)
Return $\emptyset$;
End
Calculate typicality for each of the clusters $\tau_i^k(\boldsymbol{x}_k)$ according to either the formula (23), if the distance is Mahalanobis, or according to the formula (17), if distance is Euclidean, or according to the formula (8), otherwise.
If condition (28) == 1,
Return $\emptyset$;
End
Return $\arg\max_i \tau_i^k(\boldsymbol{x}_k)$;
End
Function "Create Cluster" $(\boldsymbol{x}_k, K)$
If (the distance is Mahalanobis)
Return "Create Cluster (Mahalanobis)" $(\boldsymbol{x}_k, K)$;
End
If (the distance is Euclidean)
Return "Create Cluster (Euclidean)" $(\boldsymbol{x}_k, K)$;
End
If (the distance is any other)
Return "Create Cluster (Any)" $(\boldsymbol{x}_k, K)$;
End
End
Function "Create Cluster (Mahalanobis)" $(\boldsymbol{x}_k, K)$
$\lvert K\rvert = \lvert K\rvert + 1$;
$\mu_{\lvert K\rvert}^k = \boldsymbol{x}_k$;
If ($\lvert K\rvert = 1$)

$\Sigma_{\lvert K\rvert}^{k} = \mathbf{1}_{\text{length }(\boldsymbol{x}_k)\times\text{length }(\boldsymbol{x}_k)}$, where $\mathbf{1}_{\text{length }(\boldsymbol{x}_k)\times\text{length }(\boldsymbol{x}_k)}$ is a square matrix with all the cells equal to 0 but 1 on the main diagonal with dimensions length $(\boldsymbol{x}_k)$, i.e. the dimensionality of the space $\boldsymbol{x}_k$.
Else
$\Sigma_{\lvert K\rvert}^{k} = \text{mean}_{\text{j=[1..(K−1)]}}\left(\Sigma_i^k\right)$.
End
$K_{\lvert K\rvert}::\text{BufferForValues}\ (1) = \boldsymbol{x}_k;$
$K_{\lvert K\rvert}::\text{Support} = 1;$
End
Function "Create Cluster (Euclidean)" $(\boldsymbol{x}_k, K)$
$\lvert K\rvert = \lvert K\rvert + 1;$
$\mu_{\lvert K\rvert}^{k} = \boldsymbol{x}_k;$
If $(\lvert K\rvert = 1)$
$\sigma_{\lvert K\rvert}^{k} = 1,$
Else
$\sigma_{\lvert K\rvert+1}^{k} = \text{mean}_{\text{j=[1..(K−1)]}}\left(\sigma_i^k\right)$.
End
$K_i::\text{BufferForValues}\ (1) = \boldsymbol{x}_k;$
$K_i::\text{Support} = 1;$
End
Function "Create Cluster (Any)" $(\boldsymbol{x}_k, K)$
$\lvert K\rvert = \lvert K\rvert + 1;$
$\pi_i^k(\boldsymbol{x}) = 0;$
$K_i::\text{Support} = 1;$
$K_i::\text{BufferForValues}\ (1) = \boldsymbol{x}_k;$
End
Function "Update Clusters" $(\boldsymbol{x}_k, K_i)$
If (the distance is Mahalanobis)
Return "Update Clusters (Mahalanobis)" $(\boldsymbol{x}_k, K_i)$;
End
If (the distance is Euclidean)
Return "Update Clusters (Euclidean)" $(\boldsymbol{x}_k, K_i)$;
End
If (the distance is any other)
Return "Update Clusters (Any)" $(\boldsymbol{x}_k, K_i)$;
End
End
Function "Update Clusters (Mahalanobis)" $(\boldsymbol{x}_k,\ K_i)$
Update $\boldsymbol{\mu}_i^k$ according to the formula (15);
If $(K_i :: \text{Support} > 2)$
Update the covariance Σ_i^k according to the formulae (20), (21)
Else If $(K_i::\text{Support} == 2)$
$\Sigma_i^k = \text{cov}\ (K_i::\text{BufferForValues});$
$K_i::\text{BufferForValues} = \mathbf{1}_{0\times0};$
Else
$\Sigma_i^k = \mathbf{1}_{\text{length }(\boldsymbol{x}_k)\times\text{length }(\boldsymbol{x}_k)};$

$K_i::\text{BufferForValues}\ (K_i::\text{Support} + 1) = \boldsymbol{x}_k;$
End
$K_i::\text{Support} = K_i::\text{Support} + 1.$
End
Function "Update Clusters (Euclidean)" $(\boldsymbol{x}_k, K_i)$
Update $\boldsymbol{\mu}_i^k$ according to the formula (15);
If $(K_i::\text{Support} > 2)$
Update the variance σ_i^k according to the formula (16).
Else If $(K_i::\text{Support} == 2)$
Calculate the variance σ_i^k according to the formula (16).
Else
$\sigma_i^k = 1;$
$K_i::\text{BufferForValues}\ (K_i::\text{Support} + 1) = \boldsymbol{x}_k;$
End
$K_i::\text{Support} = K_i::\text{Support} + 1.$
End
Function "Update Clusters (Any)" $(\boldsymbol{x}_k, K_i)$
Update $\pi_i^k(x_j)$ according to the formula (7).
$K_i::\text{Support} = K_i::\text{Support} + 1.$
End

Figure 1 TEDACluster algorithm

V. TEDAClass

TEDAClass, which was previously proposed in [18], can be interpreted as a superstructure over *TEDACluster*, and requires additionally the definition of class assignment to each of the fuzzy rules.

To do this, the fuzzy rules are re-defined as

$$F = \{R_i\}, i = \overline{1,N},$$
$$R_i(\boldsymbol{x}): IF\ (\boldsymbol{x} \sim \boldsymbol{x}_i^*)\ THEN\ y_i = \boldsymbol{x}^T\Theta_i. \quad (32)$$

Here where F is a fuzzy rule system, consisting of the rules R_i, i is the index of the rule, $\boldsymbol{x} \in \mathfrak{X}, F: \mathfrak{X} \to C$, $\mathfrak{X}$ is a feature (object) space, C is a finite class ID space, $y_i \in C, \boldsymbol{x}_i^*$ is a representative point (focal point) for the cluster, Θ_i is the design matrix for the linear regression, $\sim$ is a *like* predicate, which reflects some relation of 'closeness' (association/membership) of the point $\boldsymbol{x}$ to the focal point of the fuzzy rule $R_i(\boldsymbol{x})$.

The final result is obtained as

$$\hat{y}_k = \sum_{i=1}^{N} w_i^k(\boldsymbol{x}) y_i, \quad (33)$$

The design matrix is adjusted using fuzzy RLS algorithm.

It is designed to solve the following least-squares problem [27]:

$$\left(Y_i^k - \left[\Psi_i^k\right]^T\Theta_i^k\right)^T\left(Y_i^k - \left[\Psi_i^k\right]^T\Theta_i^k\right) \to \min_{\Theta_i}, \quad (34)$$

where Y_i is the matrix of ground-truth output results for the i-th fuzzy rule, Θ_i is a design matrix, $\Psi_i = \{\boldsymbol{x}w_i^k(\boldsymbol{x})\}$ is a matrix of the RLS inputs with the membership weights $w_i^k(\boldsymbol{x})$.

The update formulae in this case are given as follows [27]:

$$\Theta_i^k = \Theta_i^{k-1} + C_i^k \Psi_i^k \left(y_i^k - \Psi_i^k \Theta_i^{k-1}\right), \quad (35)$$

$$C_i^k = C_i^{k-1} - \frac{C_i^{k-1}\Psi_i^k \left[\Psi_i^k\right]^T C_i^{k-1}}{1 + \left[\Psi_i^k\right]^T C_i^{k-1} \Psi_i^k}.$$

The final classification algorithm is given in Figure 2.

k = 1; $K = \emptyset$
While the algorithm is not terminated do
While the learning sequence is not terminated do
Wait for the new training sample x_k and its new label y_k;
Cluster_ID = Execute "Find Cluster" (x_k, K);
If (Cluster_ID is $\emptyset$)
Cluster_ID = Execute "Create Cluster" (x_k, K);
End
Execute "Update Clusters"$(x_k, K_{\text{Cluster\_ID}})$;
Execute "Update RLS regression over the cluster (classification)" $(\boldsymbol{x}_k, y_k, K_{\text{Cluster\_ID}})$
k = k + 1.
End
While the recognition sequence is not terminated do
Wait for the new recognition sample x_k and its new label y_k;
Calculate the weights $w_i^k(\boldsymbol{x_k})$ according to the formula (27).
Estimate the class $\hat{y}_k$ according to the formula (33).
k = k + 1.
End
End
Execute "Update RLS regression over the cluster (classification)" $(x_k, y_k, K_{\text{Cluster\_ID}})$
Update the regression using formula (35)
End

Figure 2 TEDAClass algorithm

VI. TEDAPredict

TEDAPredict has a structure similar to *TEDAClass*, but it is used for the prediction rather than classification.

To do this, the fuzzy rules are re-defined as

$$F = \{R_i\}, i = \overline{1, N},$$
$$R_i(\boldsymbol{x}): IF\ (\boldsymbol{x} \sim \boldsymbol{x}_i^*)\ THEN\ y_i = \boldsymbol{x}^T \Theta_i. \quad (36)$$

As before, F is a fuzzy rule system, consisting of the rules R_i, i is the index of the rule, $\boldsymbol{x} \in \mathfrak{X}, F: \mathfrak{X} \to C$, $\mathfrak{X}$ is a feature (object) space, Θ_i is the design matrix for the linear regression, $\sim$ is a *like* predicate, which reflects some relation of 'closeness' (association/membership) of the point $\boldsymbol{x}$ to the focal point of the fuzzy rule $R_i(\boldsymbol{x})$. But here $C \in \mathbb{R}^n$, $y_i \in C$, and it is used for the function approximation.

The algorithm description is given in Figure 3.

k = 1; $K = \emptyset$
While the algorithm is not terminated do
While the learning sequence is not terminated do
Wait for the new training sample x_k and its real value y_k;
Cluster_ID = Execute "Find Cluster" (x_k, K);
If (Cluster_ID is $\emptyset$)
Cluster_ID = Execute "Create Cluster" (x_k, K);
End
Execute "Update Clusters"$(x_k, K_{\text{Cluster\_ID}})$;
Execute "Update RLS regression over the cluster (regression)" $(x_k, y_k, K_{\text{Cluster\_ID}})$
k = k + 1.
End
While the regression sequence is not terminated do
Wait for the new sample x_k and its real value y_k;
Calculate the weights $w_i^k(\boldsymbol{x})$ according to the formula (27).
Estimate the regression $\hat{y}_k$ according to the formula (33).
k = k + 1.
End
End
Execute "Update RLS regression over the cluster (regression)" $(x_k, y_k, K_{\text{Cluster\_ID}})$
Update the regression using formula (35)
End

Figure 3 TEDAPredict algorithm.

VII. Experiments

A. *TEDACluster experiments*

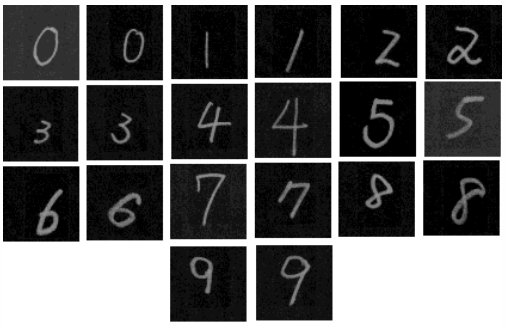

Figure 4.Experimental data samples (ETL1 database)

Figure 5 Experimental data samples (MNIST)

As an example of a complex practical problem, we considered handwritten symbol classification. As an example of this problem formulation, we used ETL1 data set [28], depicted on figure 4. For training procedure, various test sets comprised of 64 x 64 images were used, and 11031 images were selected for tests. For this data set, we calculate the cluster purity as a metric of the data division correctness. We calculate cluster purity Π for some cluster c using the following formula:

$$\Pi = \frac{1}{k}\sum_{i\in K}\max_{j\in C}\left|x_p\right|:(y_p = j \,\&\, F(x_p) = i). \quad (37)$$

Here k is a size of the data set, $1 \le p \le k$, $F(x_p): \mathfrak{X} \to K$ is a fuzzy rule system, as it was declared before, acting as an operator between the samples space $\mathfrak{X}$ and a set of cluster labels K, and y_p is a class label for the particular sample x_p.

Training set size	*Purity* Π	*Number of clusters*
50	*0.4400*	*15*
100	*0.5500*	*29*
150	*0.8000*	*82*
200	0.6400	52
300	0.7833	101
400	0.7775	121
500	0.6700	77
600	0.7200	87
700	0.7143	97
800	0.7675	132
900	0.7689	128
1000	0.7750	179
1857	0.7878	201

Table I. Clustering results for ETL1[28]

B. TEDAClass experiments

Training set size	***eClass1***	***AutoClass1***	**Neocognitron**	***TEDAClass***
500	93.26%	94.53%	92.36%	**95,18%**
1000	86.54%	95.82%	94.42%	**95,92%**
2000	96.42%	96.44%	96.04%	**96,70%**
3000	96.55%	96.50%	96.34%	**96,67%**
4000	96.62%	96.68%	96.62%	**96,88%**
5000	96.85%	96.91%	96.94%	**97,16%**
10000	97.19%	97.24%	-	**97,38%**
20000	97.32%	97.38%	-	**97,53%**
30000	97.46%	97.44%	-	**97,68%**
40000	97.45%	97.42%	-	**97,66%**
50000	97.46%	97.38%	-	**97,65%**
60000	97.46%	97.42%	-	**97,63%**

Table II Recognition results comparison for MNIST[29] database

The results in Table II reproduce the results for classification in the article [18] for MNIST dataset [29], also comprised of handwritten symbol images, normalised for a $20x20$ pixel bounding box whilst preserving aspect ratio. These results are given there to add additional interpretation to the previous TEDACluster experiments.

C. TEDAPredict experiments

For *TEDAPredict*, we used famous dataset with wine quality assessment data [30] for white wine of Portugal. This problem was considered as a regression one. We have made a comparison with the previously published results on this data set in [31][27], using the same methodology. We divide the data set randomly using 5-fold cross-validation [32] iterated 20 times and compare the results with the results published in the paper [31], where the regression was provided by multilayer perceptron neural network (NN), as well as Gaussian kernel SVM (SVM) and linear/multiple regression (MR).

The formulae for the metrics are given as follows. Let us have the unknown function $f(x), x \in \mathfrak{X}$, and its regression $\hat{f}(x)$.

Let us denote the testing data set as X_T. Then the metrics are given as follows:

$$\mathrm{MAD} = \frac{1}{|X_T|}\sum_{x\in X_T}\left|\hat{f}(x) - f(x)\right|. \quad (38)$$

$$\mathrm{A}_\alpha = \frac{1}{|X_T|}\sum_{x\in X_T}\left[\left|\hat{f}(x) - f(x)\right| \le \alpha\right]. \quad (39)$$

Here α is a tolerance threshold, $[P]$ is a predicate, which turns into 1, if it is true, and 0 otherwise.

	MR	**NN**	**SVM**	***TEDAPredict***
MAD	0.59 $\pm$ 0.00	0.58 $\pm$ 0.00	0.45 $\pm$ 0.00	0.5702 $\pm$ 0.00
$\boldsymbol{A_{0.25}}$	25.6 $\pm$ 0.1	26.5 $\pm$ 0.3	50.2 $\pm$ 0.3	29.49 $\pm$ 0.3
$\boldsymbol{A_{0.50}}$	51.7 $\pm$ 0.1	52.6 $\pm$ 0.3	64.3 $\pm$ 0.4	53.64 $\pm$ 0.4
$\boldsymbol{A_{1.00}}$	84.3 $\pm$ 0.1	84.7 $\pm$ 0.1	86.8 $\pm$ 0.2	85.15 $\pm$ 0.4

Table III. Regression results for wine dataset [30]

Here we can see, that the algorithm gives reasonable results comparing to the competitor algorithms although not better than SVM algorithms. However, in addition to the rival classifiers this one is also evolving, which gives additional advantage for real applications.

VIII. CONCLUSION

This article describes further development of the TEDAClass data mining techniques family. The results we have obtained are good enough comparing to the popular classification and regression techniques. Furthermore, the algorithm idea additionally offers an opportunity to process the data online, in evolving fashion that is especially useful for time series. The plans are to enhance the metrics which are proposed in this article, as well as make comprehensive testing on wide range of practical problems.

REFERENCES

[1] V. N. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc, 1995.

[2] C.M. Bishop. “Neural networks for pattern recognition”. Oxford: Clarendon Press(1995).

[3] P. Angelov, D. Filev and N. Kasabov (Eds.), Evolving Intelligent Systems: Methodology and Applications, 444pp., John Willey and Sons, IEEE Press Series on Computational Intelligence, April 2010.

[4] L. Rokach, O.Maimon. Data mining with decision trees: theory and applications. Vol. 69. World Scientific Pub Co Inc., 2008.

[5] Smola, A. J., and B. Schölkopf. "A tutorial on support vector regression." *Statistics and computing* 14.3 (2004): 199-222.

[6] Drucker, H., Burges, C. JC, Kaufman, L., Smola, A., and Vapnik, V.. "Support vector regression machines." *Advances in neural information processing systems* 9 (1997): 155-161

[7] McLachlan, G., and Peel, D.. *Finite mixture models*. John Wiley & Sons, 2004.

[8] Baruah, R. D. and Angelov, P. Online learning and prediction of data streams using dynamically evolving fuzzy approach. In Proceedings of the 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2013), pages 1-6, IEEE, 2013.

[9] MacQueen, J.. "Some methods for classification and analysis of multivariate observations." In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14, pp. 281-297. 1967.

[10] Angelov, P., Ramezani, R. and Zhou, X. Autonomous novelty detection and object tracking in video streams using evolving clustering and Takagi-Sugeno type neuro-fuzzy system. In IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008.

[11] Angelov, P. and Zhou, X. On line learning fuzzy rule-based system structure from data streams. In IEEE International Conference on Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence)., pages 915-922, IEEE, 2008.

[12] Angelov, P. and Yager, R. A new type of simplified fuzzy rule-based systems. In International Journal of General Systems, 41 (2): 163-185, 2012.

[13] Angelov, P., Filev, D., Kasabov, N., Cordon, O., Angelov, P., Filev, D., Kasabov, N. and Cordon, O. Evolving Fuzzy Systems. IEEE Press, 2006.

[14] Schliebs, Stefan, and Nikola Kasabov. "Evolving spiking neural network—a survey." *Evolving Systems* 4, no. 2 (2013): 87-98.

[15] Angelov, P., and Kangin, D. "Recursive SVM Based on TEDA." In Statistical Learning and Data Sciences: Third International Symposium, SLDS 2015, Egham, UK, April 20-23, 2015, Proceedings, vol. 9047, p. 156. Springer, 2015.

[16] Angelov, P. and Zhou, X. Evolving Fuzzy Rule-based Classifiers from Data Streams. In IEEE Transactions on Fuzzy Systems, 16 (6): 1462-1475, 2008.

[17] P.Angelov, D. Kangin, X. Zhou and D.Kolev. Symbol Recognition With a New Autonomously Evolving Classifier AutoClass. In 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems, pages 1-6, IEEE, 2014.

[18] D. Kangin, P. Angelov. New Autonomously Evolving Classifier TEDAClass, In IEEE Transactions on Cybernetics, 2014, submitted

[19] Zadeh, L. A. (1965). "Fuzzy sets". *Information and Control* **8** (3): 338.

[20] Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 3, No. 1, pp. 28-44, Jan. 1973.

[21] Mamdani, E.H. and Assilian, S., "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1-13, 1975.

[22] Sugeno, M. and Takagi, T., “Multi-Dimensional Fuzzy Reasoning”, Fuzzy Sets and Systems, 9-2, pp. 313-325 (1983).

[23] Angelov, P. and Yager, R. A new type of simplified fuzzy rule-based systems. International Journal of General Systems, 41 (2): 163-185, 2012.

[24] Baruah, R. D., Angelov, P. and Baruah, D. Dynamically evolving fuzzy classifier for real-time classification of data streams. Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on, pages 383-389, IEEE, 2014.

[25] E. Lughofer and E. P. Klement, “FLEXFIS: A variant for incremental learning of Takagi-Sugeno fuzzy systems, in Proc. FUZZ-IEEE 2005, Reno, Nevada, USA, 2005, pp.915-920.

[26] P. C. Mahalanobis. "On the generalized distance in statistics." *Proceedings of the National Institute of Sciences (Calcutta)* 2, 1936, pp. 49-55.

[27] P. Angelov, Evolving Takagi-Sugeno Fuzzy Systems from Data Streams (eTS+), In Evolving Intelligent Systems: Methodology and Applications (Angelov P., D. Filev, N. Kasabov Eds.), John Willey and Sons, IEEE Press Series on Computational Intelligence, pp. 21-50, ISBN: 978-0-470-28719-4, April 2010.

[28] ETL1 digits database: http://projects.itri.aist.go.jp/etlcdb/etln/etl1/etl1.htm Electrotechnical Laboratory, Japan [In Japanese and English]

[29] Y. LeCun, C.Cortes, C. J.C. Burges. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/

[30] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In *Decision Support Systems*, Elsevier, 47(4):547-553, 2009. http://www3.dsi.uminho.pt/pcortez/wine/

[31] Cortez, P., Teixeira, J., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009, January). Using data mining for wine quality assessment. In *Discovery Science* (pp. 66-79). Springer Berlin Heidelberg.

[32] Zhang, P. (1993). Model selection via multifold cross validation. *The Annals of Statistics* , 21, pp. 299-313.