Evolving Fuzzy-Rule-Based Classifiers From Data Streams

Plamen P. Angelov, *Senior Member, IEEE*, and Xiaowei Zhou, *Student Member, IEEE*

***Abstract*—A new approach to the online classification of streaming data is introduced in this paper. It is based on a self-developing (*e*volving) fuzzy-rule-based (FRB) classifier system of *T*akagi–Sugeno (*eTS*) type. The proposed approach, called *eClass* (*e*volving *class*ifier), includes different architectures and online learning methods. The family of alternative architectures includes: 1) *eClass0*, with the classifier consequents representing class label and 2) the newly proposed method for regression over the features using a first-order *eTS* fuzzy classifier, *eClass1*. An important property of *eClass* is that it can start learning "from scratch." Not only do the fuzzy rules not need to be prespecified, but neither do the number of classes for *eClass* (the number may grow, with new class labels being added by the online learning process). In the event that an initial FRB exists, *eClass* can evolve/develop it further based on the newly arrived data. The proposed approach addresses the practical problems of the classification of streaming data (video, speech, sensory data generated from robotic, advanced industrial applications, financial and retail chain transactions, intruder detection, etc.). It has been successfully tested on a number of benchmark problems as well as on data from an intrusion detection data stream to produce a comparison with the established approaches. The results demonstrate that a flexible (with evolving structure) FRB classifier can be generated online from streaming data achieving high classification rates and using limited computational resources.**

***Index Terms*—Evolving fuzzy systems, fuzzy-rule-based (FRB) classifiers, recursive least squares (RLS), Takagi–Sugeno fuzzy models.**

I. Introduction

A. Background and the State of the Art

FUZZY systems have been successfully applied to different classification tasks [1]–[7] including, but not limited to, decision making, fault detection, pattern recognition, and image processing. Fuzzy-rule-based (FRB) systems have become one of the alternative frameworks for classifier design, together with the more established Bayesian classifiers [8]–[10], decision trees [11], [12], and more recently, neural network (NN)-based classifiers [13] and support vector machines [14]. Since the classifier task is to map the set of features of sample data onto a set of class labels, fuzzy systems are particularly suitable as proven universal approximators [15]. Additionally, when compared with the NN-based classifiers, they have the advantage of greater transparency and interpretability of results [16]. Therefore, recently, the so-called neurofuzzy hybrids have become popular [5]. Originally, FRB classifiers (similar to FRB systems) were designed based on linguistic and expert knowledge. Since the 1990s, however, the so-called data-driven approaches have become dominant in the fuzzy systems design area [5], [28]. Note that data-driven methods can preserve and even improve the interpretability of the fuzzy system [7], [16]–[18].

In this paper, we use FRB classifiers as a promising linguistic framework that offers high performance (classification rate) and computational efficiency. Another important reason for using this type of classifier is that it is suitable for the realization of incremental online and evolving schemes.

B. Need for Evolving Online Classifiers

We are witnessing an information revolution; currently, large quantities of information are produced at a fast rate by the sensors in advanced industrial processes, autonomous systems, space, and aircraft, by the users of the Internet, the finance industry, consumer markets, etc. The challenges that are faced by information processing, and classification in particular, are related to: 1) the need to cope with huge amounts of data and 2) process streaming data online and in real time [20], [21]. Storing the complete dataset and analyzing the data streams in an offline (batch) mode is often impossible or impractical, and data streams are very often nonstationary. At the same time, most of the conventional classifiers are designed to operate in batch mode and do not change their structure online (do not capture new patterns that may be present in the data stream once the classifier is built). An example from the network intrusion detection area is the arrival of new, previously unseen threats (hackers are very inventive and they use newer ways to disrupt the normal operation of servers and users). Offline pretrained classifiers may be good for certain scenarios, but they need to be redesigned or retrained for new circumstances. The conventional classifiers applied to *data streams* extract a "snapshot" of the data stream and require all the previous data, which implies higher memory and computational requirements. This is in addition to the classical pair of requirements, namely: 1) high classification rate and 2) simple and interpretable/transparent classifier structure [16], [17]. In contrast, the so-called *incremental* (or online) classifiers work on a per-sample basis and only require the features of that sample plus a small amount of aggregated information (a rule base, a small number of variables needed for recursive calculations); but they do *not* require *all the history* of the data stream (all previously seen data samples). Sometimes they are also called *one-pass* (each sample is processed only once at a time and is then discarded from the

Manuscript received February 2, 2007; revised June 26, 2007, October 15, 2007, and December 11, 2007; accepted December 19, 2007. First published May 23, 2008; current version published December 19, 2008.

The authors are with the Intelligent Systems Research Laboratory, Department of Communication Systems, Infolab21, Lancaster University, Lancaster LA1 4YW, U.K. (e-mail: p.angelov@lancaster.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2008.925904

memory). To address the challenge to classify *streaming* data online and in real time with a classifier that adapts/evolves its structure, we introduce the *eClass* family.

C. Evolving Versus Genetic/Evolutionary Classifiers

Genetic and evolutionary algorithms [22], [23] are computational techniques for a "directed" random search for a solution to a loosely formulated optimization problem, which borrows from nature the concept of *genetic* or *population* evolution based on computational mechanisms that mimic such paradigms as *mutation*, chromosomes *crossover, reproduction*, and *selection*. The definition that the Oxford Dictionary [24, p. 358] gives to "*genetic*" is a "branch of biology dealing with the heredity, the ways, in which characteristics are passed on from parents to offspring." Similarly, "*evolutionary*" [24, p. 294] is the "development of more complicated forms of life (plants, animals) from earlier and simpler forms." Genetic/evolutionary algorithms have been applied successfully as learning techniques for the offline design of FRB classifiers [1], [2] and decision trees [25].

Meanwhile, FRB systems that are *evolving* ("*gradually developing*") have recently been introduced [26] and successfully applied to a range of problems, such as online system identification [27], time series prediction [28], fault detection [29], intelligent sensors [30], and control. The definition of *evolving* in [24, p. 294] is "unfolding; developing; being developed, naturally and gradually." While *genetic/evolutionary* is related to the population of individuals and parents-to-offspring heredity, *evolving* is applicable to individuals' self-development (known in humans as autonomous mental development). "*Evolving*" relates more to *learning from experience*, *gradual* change, knowledge generation from routine operation, and rules extraction from the data. Such capabilities are highly demanded by autonomous, robotic, and advanced industrial systems, among others. While a *genetic/evolutionary* FRB system generates new rules as a *crossover* or *mutation* of previous rules driven by a "directed" random process [1], [2], an *evolving* FRB system learns new rules from new data *gradually* preserving/inheriting the rules learned already. This is very similar to the way in which individual people (especially children) learn. In a similar way to a human, an evolving fuzzy system can be initiated by an initial rule base or can start learning "from scratch." This paradigm has been applied to NNs [31] and fuzzy systems [26]–[30]. In this paper, we propose *e*volving FRB *class*ifiers (*eClass*), which are characterized by the "*gradual* development" of their structure from the streaming data.

D. Evolving Versus Incremental

In the data mining literature, there are a number of approaches to address the problems of *data streams* [10]–[12], [32], [33], [35]–[39]. Incremental classifiers are one of them. They have been implemented in different frameworks: decision trees [32], NN-based [e.g., adaptive resonance theory (ART) [34], incremental learning vector quantizers (iLVQs)], probabilistic [e.g., incremental versions of Bayesian classifiers [10], linear discriminant analysis (iLDA) [33], principal component analysis (iPCA) [33]. Note, however, that the classifier *structure* in these cases is assumed to be *fixed*. In terms of data streams, such classifiers cannot address the problem of the so-called concept *drift* and *shift*. By *drift* in the machine learning literature, they refer to a modification of the concept over time [38], [39]. This relates to a rather smooth transition of the data distribution from one local region of the feature space (described in a fuzzy classifier by a fuzzy rule) to another. Very often, however, there are concept *shifts* [38], [39]. This usually has some practical meaning, such as the sudden appearance of a fault or an abrupt change of a regime of operation [29]. To capture such sudden and abrupt changes, one would require not only tuning parameters of the classifier, but a *change in its structure* as well. To the best of our knowledge, the FRB classifier, as proposed in this contribution, is the first one to have an *evolving structure* that can be seen as a technique to address the concept *drift* to new areas of the data space and concept *shift*. NNs with *evolving structure* have also been proposed recently [31], but so far they have only been applied to time series prediction and not to classification.

E. Structure of the Paper

The remainder of the paper is organized as follows. Section II describes the architecture of *eClass0*, which is an FRB classifier with class labels as outputs. Section III details a more effective scheme, *eClass1*, which performs regression over the features. Section IV describes the experimental results based on a number of widely used benchmark datasets organized as pseudostreams, as well as using intruder detection data [40]. Finally, Section V presents our conclusions. The derivation of the recursive cosine-distance-based potential and pseudocode of the proposed algorithms are detailed in the Appendix.

II. eClass0

A. Architecture of eClass0

A classifier is a mapping from the feature space to the class label space. An FRB classifier describes, with its antecedents part, the fuzzy partitioning of the feature space $x \in R^n$, and with the consequent part, the class label $L_i, i = [1, K]$. The structure of *eClass0* follows this typical construct of an FRB classifier [1]–[4]:

$$R^i: \text{IF } \left(x_1 \text{ is around } x_1^{i*}\right) \quad \text{AND} \quad \left(x_2 \text{ is around } x_2^{i*}\right) \quad \text{AND} \quad \cdots \quad \text{AND} \quad \left(x_n \text{ is around } x_n^{i*}\right) \quad \text{THEN } \left(L^i\right) \tag{1}$$

where $x = [x_1, x_2, \ldots, x_n]^T$ is the vector of features, R^i denotes the ith fuzzy rule, $i = [1, N]$, N is the number of fuzzy rules, $\left(x_j \text{ is around } x_j^{i*}\right)$ denotes the jth fuzzy set of the ith fuzzy rule, $j = [1, n]$, x^{i*} is the focal point of the ith rule antecedent (note that this is a prototype—a real, existing data sample and not the mean), and L^i is the label of the class of the ith prototype (focal point).

The inference in *eClass0* is produced using the well-known "winner takes all" rule [1]–[4], [42]:

$$L = L^{i*}, \qquad i^* = \arg\max_{i=1}^{N} \left(\tau^i\right) \tag{2}$$

where τ^i denotes the firing level of (degree of confidence in) the ith fuzzy rule, which is determined as a product (a t-norm that represents the logical AND operator [2], [18]) of the membership values μ_j^i of the jth feature, $j = [1, n]$, and the fuzzy set $\left(x_j \text{ is around } x_j^{i*}\right)$:

$$\tau^i = \prod_{j=1}^{n} \mu_j^i(x_j), \qquad i = [1, N]. \tag{3}$$

The membership functions that describe the degree of association with a specific prototype are of Gaussian form (characterized by good generalization capabilities and coverage of the whole feature space):

$$\mu_j^i = \exp\left(-\frac{1}{2}\left(\frac{d_j^i}{r_j^i}\right)^2\right), \qquad i = [1, N], \qquad j = [1, n] \tag{4}$$

where d_j^i is the distance between a sample and the prototype (focal point) of the ith fuzzy rule and r_j^i is the spread of the membership function, which also represents the radius of the zone of influence of the fuzzy rule.

Note that this representation resembles the normal distribution [8], [9], and the spread of the membership function can also be represented by the standard deviation. The spread r_j^i is determined based on the scatter σ_j^i [43] of the data *per cluster/rule* ($i = [1, N]$) and *per feature* ($j = [1, n]$)

$$\sigma_{jk}^i = \sqrt{\frac{1}{S_k^i}\sum_{l=1}^{S_k^i} d^2(x_j^l, x_j^{i*})}, \qquad \sigma_{j1}^i = 1 \tag{5}$$

where S_k^i denotes the *support* of the ith cluster/rule at the kth time instant (when k data samples are read).

Note that in (4) and (5), the projection of the distance d_j^i on the axis formed by the jth feature is used. This leads to hyperellipsoidal clusters with different spreads for different features (see also Fig. 1) and facilitates the interpretation. *Support* is the number of points that are in the zone of influence of a cluster [43]. It is initialized with 1 when a prototype is generated and incrementally increases by 1 afterward for each sample that is closer to that prototype than to any other:

$$S_{k+1}^i = S_k^i + 1, \quad i = \arg\min_{i=1}^{N} \left\|x_k - x^{i*}\right\|, \quad k = 2, 3, \ldots. \tag{6}$$

When a new cluster/rule is formed, $N \leftarrow N + 1$, its initial scatter is approximated by the average of the scatter for the existing fuzzy rules for that class [43], and its support is initially set to 1:

$$\sigma_k^{N+1} = \frac{1}{N}\sum_{i=1}^{N} \sigma_k^i, \qquad S_k^{N+1} \leftarrow 1, \qquad k = 2, 3, \ldots. \tag{7}$$

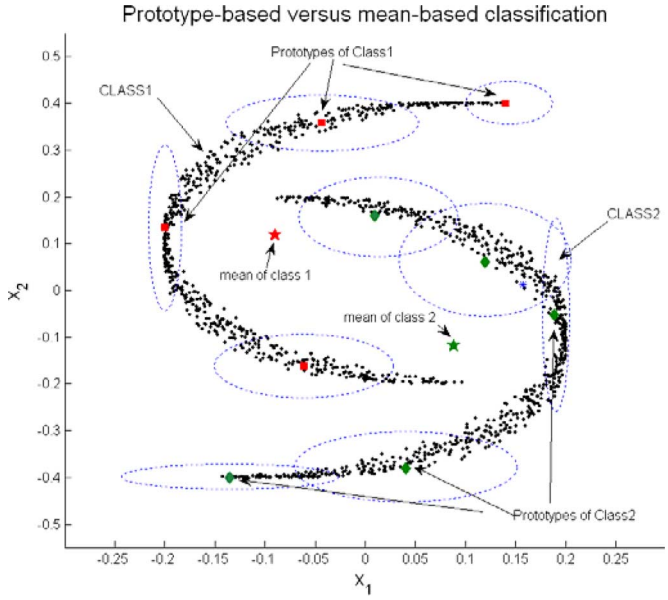


Fig. 1. Prototype-based classification versus mean-based classification.

Here, we also introduce *support per class* Q_k^l, which represents the number of data samples that are associated with the clusters that form the antecedent parts of all fuzzy rules that have the same consequents N^l:

$$Q_k^l = \sum_{j=1}^{N^l} S_k^j. \tag{6a}$$

The distance d in (5) can have a Euclidean [26]–[28] or a *cosine* form. The cosine distance can cope with problems, such as a different number of features and zero values, and is defined as [9]

$$d_{\cos}(x_k, x_l) = 1 - \frac{\sum_{j=1}^{n} x_{kj} x_{lj}}{\sqrt{\sum_{j=1}^{n} x_{kj}^2 \sum_{j=1}^{n} x_{lj}^2}} \tag{8}$$

where x_k and x_l are two n-dimensional vectors.

B. Learning eClass0

Typically, FRB classifiers are trained offline using evolutionary algorithms [1]–[4], [41], [42] or gradient-based schemes, such as backpropagation when combined with NN [5]. The *eClass* family, however, is designed for online applications with an evolving (self-developing) FRB structure. To achieve this, the antecedents of the FRB are formed from the data stream around highly descriptive focal points (prototypes) in the input–output space $z = \left[x^T, L\right]^T$ *per class*. In this paper, we modify the recently introduced recursive density-based fuzzy partitioning mechanism [44] that is very robust to outliers and noise so that it works *per class* and uses *cosine* distance. Also, we apply this online evolving (open structure) partitioning to a classification problem while it is originally introduced [27], [44] as a first stage of a predictive models design. This data partitioning mechanism identifies the most representative prototypes and builds fuzzy rules around them. In this way, it leads

to the formation of information granules, described linguistically by fuzzy sets. Thus, it serves the transformation of data into primitive forms of knowledge. This online algorithm works similarly to adaptive control and estimation [45]—in the period between two samples, two phases are performed: 1) class prediction (classification) and 2) classifier update or evolution. During the first phase, the class label is not known and is being predicted; during the second phase, however, it is known and is used as supervisory information to update the classifier (including its *structure evolution* as well as its parameters update). The second phase may be active for certain data samples only.

The main difference between *eClass0* and a conventional FRB classifier is: 1) the open structure of the rule base (it self-develops online starting "from scratch," while in a conventional FRB classifier, it is determined offline and then fixed) and 2) the online learning mechanism that takes into account this flexible rule-base structure. Note that the overall FRB is composed of K subrule bases so that, in each subrule base, the consequents of all rules are the same, but the number of rules N should be no less than the number of classes ($N \geq K$). That is, every new data sample with a class label that has not been previously seen becomes automatically a prototype. Since there is a prototype replacement and removal mechanism, this is usually temporary (this prototype is often later replaced by more descriptive prototypes). In this way, the classifier learns "from scratch" and the number of classes does not need to be known in advance.

The focal points of the fuzzy rules are generated as a result of a projection of Gaussian membership functions centered at prototypes onto the axes representing different features. This process is widely used in data-driven approaches [41] for fuzzy systems design and is facilitated by the use of standard Euclidean or cosine distance measures. The difference from most of the other classification approaches is that fuzzy sets in *eClass* are formed around existing data samples (prototypes), not around means (Fig. 1). Note that the latter can often be infeasible, as the means are usually nonexisting abstract points.

The basic notion of the partitioning algorithm is that of the *potential*, which is defined as a Cauchy function of the sum of distances between a certain data sample and *all* other data samples in the feature data space [44]. The interpretation of the *potential* is as a measure of the data *density*

$$P_k(z_k) = \frac{1}{1 + \left(\sum_{i=1}^{k-1} d(z_k, z_i)\right) \Big/ (k-1)} \tag{9}$$

Equation (9) defines the so-called *global* potential [43]. In this paper, we define *local* (*per class*) potential as a measure of the data density for that specific class, l:

$$P_k^l(z_k) = \frac{1}{1 + \left(\sum_{i=1}^{Q_k^l - 1} d(z_k, z_i)\right) \Big/ (Q_k^l - 1)}. \tag{10}$$

Partitioning using the *potential* is based on the following principle: "The point with the highest potential is chosen to be the focal point of the antecedent of a fuzzy rule" [46], [47]. In this way, fuzzy rules with high descriptive power and generalization capabilities are generated. This is illustrated in Fig. 2.

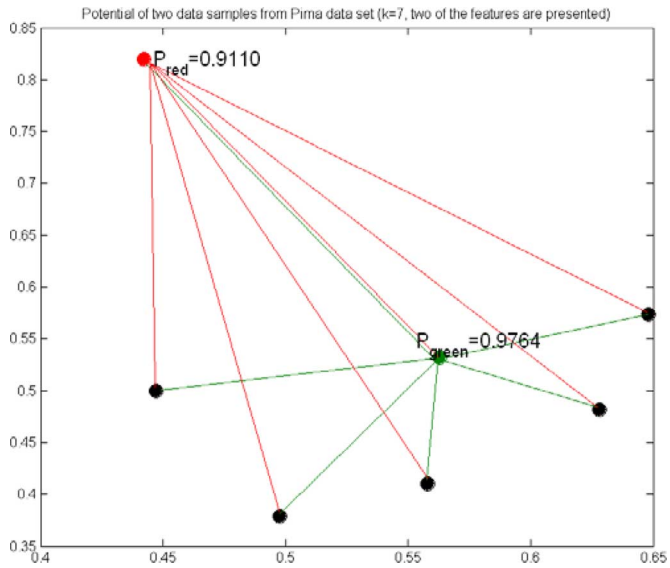


Fig. 2. Potential of data samples (Pima dataset, two of the features, $k = 7$).

Although we used Euclidean distance in [26], in the present paper, we develop the *recursive* formulas for calculating the potential when *cosine* distance is used. The *recursive* calculation that is introduced (the detailed derivation is given in the Appendix) needs the current data point z_k^j and (n + 1) memorized quantities only (β_k and $\chi_k^j,\ j = [1, n]$)

$$P_k^l(z_k) = \frac{1}{2 - \left[1/\sqrt{\sum_{j=1}^{n+1}\left(z_k^i\right)^2}\right] \sum_{j=1}^{n+1} z_k^i b_{k-1}^i}, \qquad k = 2, 3, \ldots, \qquad P_1(z_1) = 1 \tag{11}$$

where

$$b_k^i = b_{k-1}^i + \sqrt{\frac{\left(z_k^i\right)^2}{\sum_{j=1}^{n+1}\left(z_k^j\right)^2}}, \qquad b_1^i = \sqrt{\frac{\left(z_1^i\right)^2}{\sum_{j=1}^{n+1}\left(z_1^j\right)^2}}, \qquad i = [1, n+1].$$

Each time a new data sample is read, it affects the data density; therefore, the potentials of the existing centers need to be updated. This update can also be done in a *recursive* way (the derivation is also given in the Appendix):

$$P_k^l\left(z_l^{i*}\right) = \frac{\left(Q_k^l - 1\right) P_{k-1}^l\left(z_l^{i*}\right)}{Q_k^l - 1 + \left[\left(Q_k^l - 2\right)\left(\frac{1}{P_{k-1}^l\left(z_l^{i*}\right)} - 1\right) + d_{\cos}\left(z_l^{i*}, z_k\right)\right]}, \qquad k = 2, 3, \ldots. \tag{12}$$

Once the potential of the new coming data sample is calculated *recursively* using (11) and the potential of each of the previously existing prototypes is recursively updated using (12), they are compared. If the new data sample has a *higher* potential than any of the previously existing prototypes of that class, L, then it is a good candidate to become a focal point of a new rule

in this subrule base because it has high descriptive power and generalization potential:

$$P_k^L(z_k) > P_k^L\left(z_L^{i*}\right) \qquad \forall i^* \in N^L. \tag{13}$$

Forming a new fuzzy rule around the newly added prototype leads to a *gradual* increase of the size of the subrule base, which is why the approach is called "*evolving*":

$$z^{(N^L+1)*} \leftarrow z_k. \tag{14}$$

The potential of the newly generated rule is set to 1 temporarily (it will be updated to take into account the influence of each new data sample on the generalization potential of this particular focal point by (12) with each new data sample being read):

$$P_k^L\left(z^{(N^L+1)*}\right) \leftarrow 1. \tag{15}$$

To increase the interpretability and update the rule base, one needs to remove previously existing rules that become ambiguous after the insertion of a new rule. Therefore, each time a new fuzzy rule is added, it is also checked whether any of the already existing prototypes in this subrule base is described by this rule to a degree higher than e^{-1} (this is an analogy to the so-called one-sigma condition [9]):

$$\exists i, \quad i = [1, N^L], \quad \mu_i^j(z^{N^L+1}) > e^{-1} \quad \forall j, \quad j = [1, n]. \tag{16}$$

If any of the previously existing prototypes satisfies this condition, the rules that correspond to them are removed from this subrule base (in fact, replaced by the newly formed rule).

The spreads of the membership functions of the subrule base of the respective class are also recursively updated based on the data distribution [48]:

$$r_k^i = \rho r_{k-1}^i + (1-\rho)\sigma_{k-1}^i, \qquad i = [1, N^L] \tag{17}$$

where ρ is a learning parameter; it determines how quickly the spread of the membership functions will converge to the local scatter of that cluster. The default value of 0.5 further simplifies the expression into

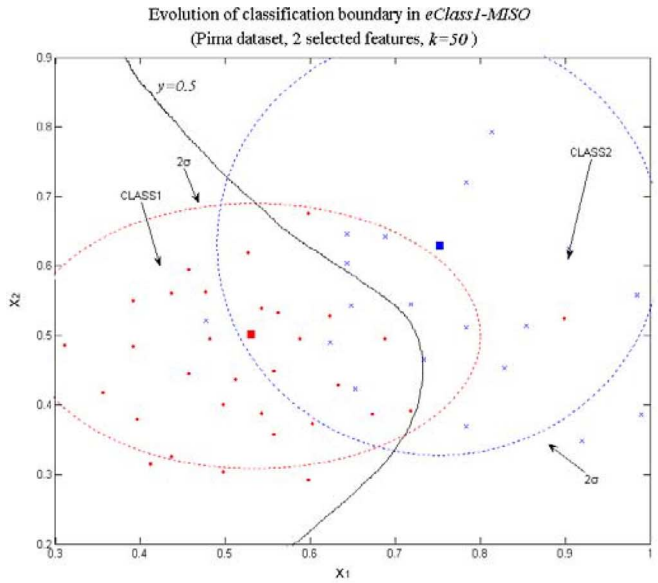
$$r_k^i = \frac{r_{k-1}^i + \sigma_{k-1}^i}{2}, \qquad i = [1, N^L]. \tag{18}$$

The procedure of *eClass0* is summarized in the pseudocode given in the Appendix.

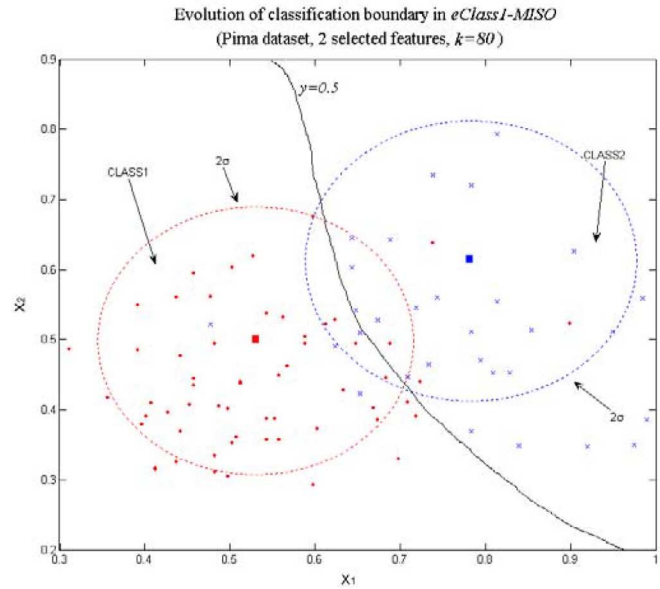
III. eClass1

A. Architecture of eClass1

The architecture of *eClass1* differs significantly from the architecture of *eClass0* and the typical FRB [1]–[5], [41], [42] in that it regresses over the feature vector using first-order multiple-input–multiple-output evolving Takagi–Sugeno (*MIMO-eTS*) models [49]. Note that the classification surface in a data stream is dynamically changing (see Fig. 3), and *eClass1* aims to evolve its rule base so that it reacts to this by dynamically adapting parameters of the classifier (spreads, consequent parameters) as well as the focal points and the size of the rule base. *eClass1* is



(a)



(b)

Fig. 3. Evolution of classification boundary in *eClass1-MISO* (pima dataset, two of the features) (a) after $k = 50$ data samples are read, and (b) after $k = 80$ data samples are read.

defined by a set of rules of the following type:

$$R^i: \text{IF}\left(x_1 \text{ is around } x_1^{i*}\right) \text{ AND } \cdots \text{ AND } \left(x_n \text{ is around } x_n^{i*}\right) \text{ THEN } \left(y^i = \overline{x}^T\Theta\right) \tag{19}$$

where $\overline{x} = [1, x_1, x_2, \ldots, x_n]^T$ denotes the $(n+1)\times 1$ extended vector of features and y^i is the output.

The outputs of particular rules can be normalized so that they sum up to 1 by

$$\overline{y}^i = \frac{y^i}{\sum_{i=1}^N y^i}. \tag{20}$$

One can easily check that $\sum_{i=1}^N \overline{y}^i = 1$.

The normalized outputs per rule can be interpreted [56] as the *possibility* of a data sample belonging to a certain class if we use target labels for the classes: 0 for nonmembership and 1 for membership. In that respect, this approach has some similarity to the so-called indicator matrix [8] used in offline crisp classifiers. Note that a conventional FRB classifier has a *class label* as outputs of each rule. In the case that *MISO eTS* is used (for *two*-class problems), outputs per fuzzy rule are scalars and the approach resembles a fuzzily weighted locally valid LDA. In this case, parameters of the local linear models Θ are vector column $\Theta = \begin{bmatrix}\theta^i_{01} & \theta^i_{11} & \cdots & \theta^i_{n1}\end{bmatrix}^T$. In the case that *MIMO eTS* is used, they can form an $(n+1) \times K$ matrix

$$\Theta = \begin{bmatrix} \theta^i_{01} & \theta^i_{02} & \cdots & \theta^i_{0K} \\ \theta^i_{11} & \theta^i_{12} & \cdots & \theta^i_{1K} \\ \cdots & \cdots & \cdots & \cdots \\ \theta^i_{n1} & \theta^i_{n2} & \cdots & \theta^i_{nK} \end{bmatrix}.$$

In this case, the outputs of the fuzzy rules form a *K*-dimensional row vector—one normalized output for each class, $\overline{y}^i = [\overline{y}^i_1, \overline{y}^i_2, \ldots, \overline{y}^i_K]$.

The overall output in *eClass1* is formed as a weighted sum of the normalized outputs of each rule, also called the "center-of-gravity" [50] principle, which is typical for regression, time series prediction, and modeling, but not for classification:

$$y = \sum_{i=1}^{N} \frac{\tau^i}{\sum_{j=1}^{N} \tau_j} \overline{y^i.} \tag{21}$$

The output is then used to discriminate between the classes. In a *two-class* problem using *MISO eTS*, one can have targets of 0 for class 1 and 1 for class 2, or vice versa. The discrimination in this case will be at a threshold 0.5 with outputs above 0.5 classified as class 2 and outputs below 0.5 classified as class 1, or vice versa. In problems with more than two classes, one can apply *MIMO eTS*, where each of the *K* outputs corresponds to the possibilities of the data sample to belong to a certain class (as discussed before). Note that one can use *MIMO eTS* for a *two*-class problem, too. Then the target outputs will be 2-D, e.g., $y = [1\ 0]$ for class 1 and $y = [0\ 1]$ for class 2 or vice versa. The label for *eClass1-MISO* (which is used in *two*-class problems) is determined by

$$\text{IF } (y > 0.5) \quad \text{THEN } (Class1) \quad \text{ELSE } (Class2). \tag{22}$$

Fig. 3 illustrates the classification of a benchmark (Pima [51]) dataset into two classes (signs of diabetes or no signs of diabetes) after 50 and after 80 data samples. The two fuzzy rules formed are represented by their focal points in a 2-D illustration for two of the nine features. The nonlinear classification surface (the solid curve in Fig. 3) is derived for this *two*-class problem as $y = 0.5$. This figure illustrates the dynamics when comparing the two plots (after 50 samples are read, and after 80 samples), which illustrates the need for online and evolving classification schemes.

In *eClass1-MIMO*, the label is determined by the highest value of the discriminator $\overline{y}_l$:

$$L = L_{i*}, \qquad i^* = \arg\max_{l=1}^{K} \overline{y}_l. \tag{23}$$

Note that (23) is not the typical "winner takes all" principle in terms of the firing strength of the rules as applied in classification problems. In this sense, the proposed approach resembles more the LDA [9] and support vector machine [14], rather than typical fuzzy classifiers [1]–[5], [7]. Also note that a *two*-class problem can be solved by *eClass1-MISO* and by *eClass1-MIMO*, but while the number of parameters of the antecedent part in both architectures is the same, the number of consequent parameters in *eClass1-MISO* is halved ($n+1$ since $\Theta = [\theta^i_{01} \quad \theta^i_{11} \quad \cdots \quad \theta^i_{n1}]^T$) compared to *eClass1-MIMO*, i.e., $2 \times (n+1)$, since

$$\Theta = \begin{bmatrix} \theta^i_{01} & \theta^i_{11} & \cdots & \theta^i_{n1} \\ \theta^i_{02} & \theta^i_{12} & \cdots & \theta^i_{n2} \end{bmatrix}^T.$$

In *eClass1*, the quality of the fuzzy rules is constantly monitored by calculating their "*age*." The *age* of a fuzzy rule [43] is especially important for data streams. It gives accumulated information about the timing of *when* a certain sample was assigned to a cluster/respective, rule. It is well known that incremental approaches are order-dependent. With a rule's *age*, one can make use of this specific feature of the data streams. It is proposed that *age* is calculated as

$$A^i = k - \frac{\sum_{l=1}^{S^i_k} k_l}{S^i_k} \tag{24}$$

where k_l is the time index when the data sample was read.

Note that the factor $\sum_{l=1}^{S^i_k} k_l / S^i_k$ can easily be calculated recursively, and thus, *A* itself can be calculated recursively. When a new data sample creates a new rule, its *age* is initiated by the index of that sample (the time instant, if in a time-related stream). Each time a new data sample is simply assigned to an existing rule, the *age* of that rule gets smaller [see (24)]. If no sample is assigned to a rule, it gets older by 1. Note that the range of *A* is [0; *k*]. The *age* of fuzzy rules (and *aging rate*—a derivative of *age* in terms of the sampling period *k*) proved to be very useful in the online analysis of the concept *drift* in the data stream. While the classifiers of the *eClass* family aim to react to the concept *shift* by evolving their structure online, generating new rules and removing outdated rules, the online analysis of the (first and second) gradient of the *age* of the rules can identify the concept *drift*. In Fig. 4, we illustrate this for the example of an ionosphere dataset from the University of California, Irvine (UCI), repository [51].

The first image [Fig. 4(a)] illustrates the case where the rules are *aging* with a normal rate (the gradient of the *age* is smaller but close to 1, and more importantly, it is nearly constant). The second image [Fig. 4(b)] illustrates a case where there are periods (around samples 155 and 213) where the data cause a strong update of a specific rule (no. 3). In this case, the gradient of the *age* decreases and even becomes negative, and then, its *age* increases again (the rule is not updated so often or not at

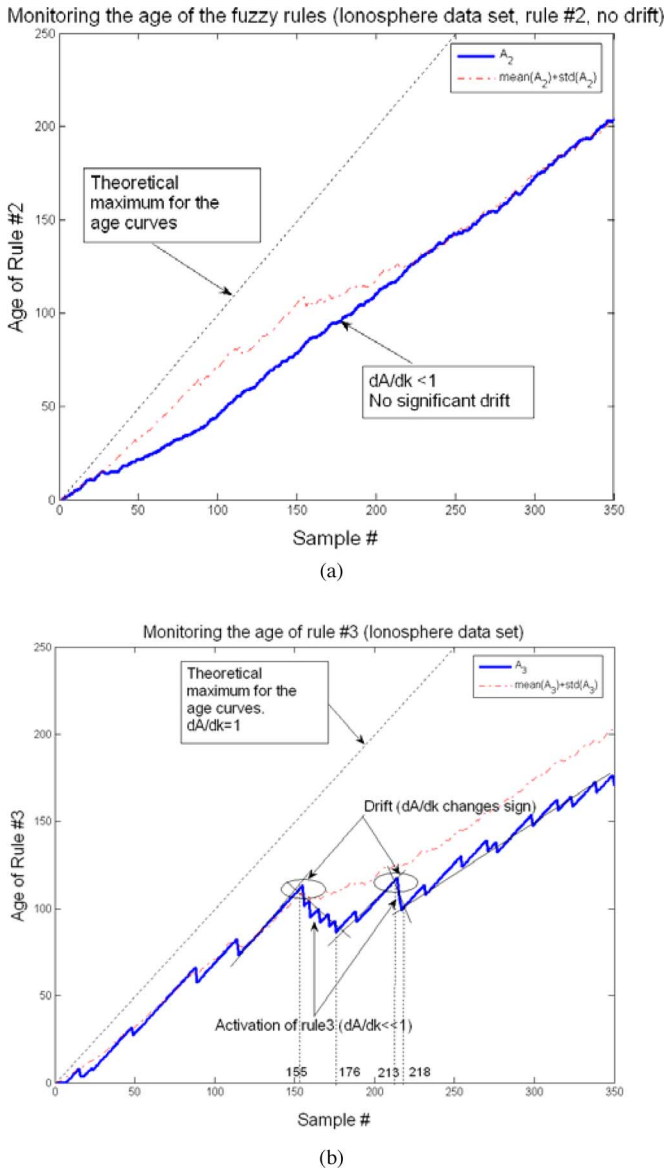


Fig. 4. Age evolution (ionosphere dataset). (a) Rule 2, no *drift*. (b) Rule 3, *drift detected*.

all). The explanation is that when a concept *drift* occurs (as in the second image), the frequency of usage of some rules (as for rule 3) will decrease because a transition from one operating state, which affects the data density in one local region (around the focal point of rule 3), to another one, which affects the data density in another local region, takes place.

The following simple rule for updating of the rule base is introduced by *removing older* rules (rules whose *age* is above the mean *age* for that rule by more than the "one-sigma" value, calculated recursively up to that moment):

$$\text{IF } (A^i > \overline{A^i} + \text{std}(A^i)) \quad \text{THEN } (\lambda^i \leftarrow 0; N \leftarrow (N-1)) \tag{25}$$

where $\overline{A^i}$ denotes the mean *age* and $\text{std}(A^i)$ denotes the standard deviation of the *age* of the ith rule.

B. Learning eClass1

Learning for *eClass1* is based on the approach for online learning TS models with evolving structure introduced in [27] and extended for the MIMO case in [49]. Learning is based on the decomposition of the identification problem into: 1) FRB structure design and 2) parameters identification. Both of these subproblems can be performed in online mode during a single time step (per sample).

The first subproblem, structure identification, is addressed using scatter-based fuzzy partitioning as described for *eClass0*. The main difference is that, in *eClass1*, the potential is calculated *globally and not per class*. The reason is that the aim of partitioning in *eClass0* is different from that in *eClass1*. In *eClass0*, it is performed with the aim of identifying representative prototypes, which have *high within-class density*. Note that this is within-class, but not necessarily within-cluster (per rule) density and is expressed mathematically by the local potential [see (9)]. In *eClass1*, however, clustering serves quite a different role—it is one of the two phases of the TS model identification and is combined with the consequents parameter identification using a version of the recursive least squares (RLS) technique. A similar approach is used for offline TS fuzzy systems designed in [47] and for evolving online TS (*eTS*) in [27]. In this paper, we use the online learning of *eTS* to regress on the features in order to estimate the possibility of a data sample belonging to a certain class. Since, in *eClass1*, we have one FRB that is *not* divided *per class*, the *global* potential is used to identify the prototypes that are representative for *any* class. It is recursively calculated by (11) but applied *globally*, not *per class*. The equation for the update of the potential of all previous prototypes each time a new data sample is read is given by:

$$P_k\left(z^{i*}\right) = \frac{(k-1)\,P_{k-1}\left(z^{i*}\right)}{k-1+\left[(k-2)\left[\{1/P_{k-1}(z^{i*})\}-1\right]+d_{\cos}\left(z^{i*},z_k\right)\right]}, \quad k=2,3,\ldots. \tag{12a}$$

Then, the potentials of the new data sample (11) and the updated potential of all of the previous focal points are compared by applying (13). The algorithm then follows the logic of the algorithm described in Section II-B with the only differences being that the potential is *global* instead of *local* and N^1 is replaced by N in all equations.

Once the antecedents are formed, the consequents parameters estimation (the second subproblem of the design of *eClass1*) is addressed as a fuzzily weighted RLS estimation problem *per rule* [27]:

$$\Theta_k^i = \Theta_{k-1}^i + C_k^i \lambda^i \overline{x}_k (y_k - \overline{x}_k^T \Theta_{k-1}^i), \qquad \Theta_1^i = 0 \tag{26}$$

$$C_k^i = C_{k-1}^i - \frac{\lambda^i C_{k-1}^i \overline{x}_k \overline{x}_k^T C_{k-1}^i}{1+\lambda^i \overline{x}_k^T C_{k-1}^i \overline{x}_k}, \quad C_1^i = \Omega I, \quad k=2,3,\ldots \tag{27}$$

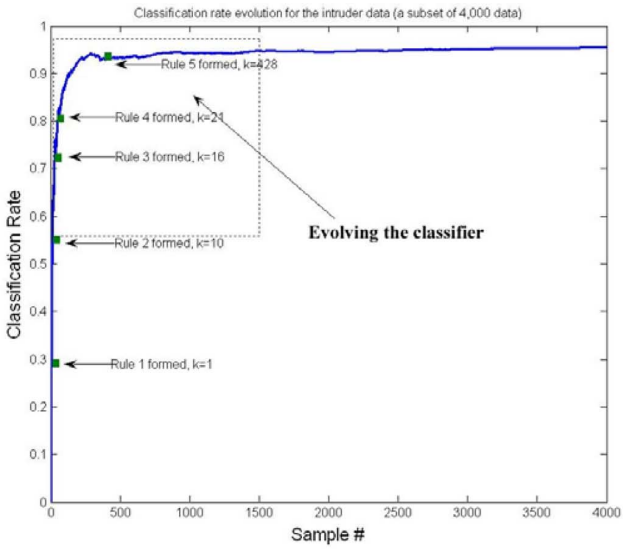
where $C \in R^{N(n+L)xN(n+L)}$ denotes the covariance matrix, Ω is a large positive number, and I is the identity matrix.

eClass1 is optimally (in LS sense) [27], [48] tuned in terms of consequents parameters Θ. In terms of its antecedents and rule-base structure, it is based on the robust online partitioning approach [44]. Condition (13) is very hard to satisfy, and therefore, usually a small number of rules are formed; additionally, condition (16) further automatically removes rules with ambiguous meaning. The objective of the optimal estimation [seen from the factor in brackets in (26)] is to generate values as close as possible to 1 for the samples of certain classes and values as close to 0 for all other classes by regressing on the feature space [see (19)]. In this way, *eClass1* is unique as an evolving robust nonlinear FRB classifier suitable for the online applications required in data streams. The algorithm procedure is given in the Appendix.

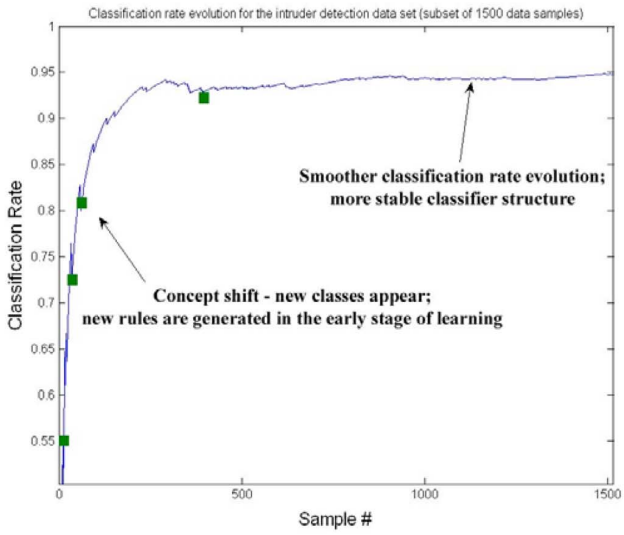
The main novelties of *eClass1* can be summarized as follows.

1) It is evolving (the rule base is not pretrained and fixed; learning can start "from scratch" with the very first data sample).
2) It can have a *MIMO* structure and thus build a separate regression model for each class (output). Note that if a sample with a previously unseen class label is met, the *MIMO* structure expands naturally by initializing learning of the new $(L+1)$th class from this point onward in the same way as for the remaining L classes.
3) It can attain high classification rates compared favorably to well-known offline and incremental classifiers (as seen in the next section).
4) It is one-pass, recursive, and therefore, has extremely low memory requirements.
5) It is useful for online analysis and monitoring of the concept *drift* using fuzzy rules *aging* (see Fig. 4).

The effect of gradual evolution in learning is illustrated in Fig. 5, where the same validation data have been used after different periods of incremental and evolving learning for the example of intrusion detection data. It can be seen that the classification rate improves when more data are used for evolving the classifier online. The increase in the classification rate is not perfectly smooth [as can be seen in the zoomed area in Fig. 5(b)], as the new data samples bring noise as well as useful information. Also, as new classes are met for the very first time, the classification rate drops for a short period. In *eClass1*, the new data samples are used not only to form new prototypes, but also to update the consequents parameters by (26) and (27), in order to adapt to the dynamically changing nonlinear classification surface (Fig. 3). It is obvious, however, that an overall stabilization of the classification rate to values well over 90% based on only four fuzzy rules [the time stamp of which is shown in Fig. 5(a)] occurs after a very small number of samples—just over 100 out of a total dataset of 5 000 000 samples. Similar behavior has been observed for the other experimental cases.



(a)



(b)

Fig. 5. Evolution of the classification rate during online learning of *eClass1* with a subset of intruder detection dataset. (a) Subset of 4000 data samples from the four subclasses; the square dots indicate the time instant of the generation of a new fuzzy rule (rule base evolution). (b) Zoom-in of the previous plot for 1500 data.

IV. Experimental Results

The classifiers from the *eClass* family were tested on a number of widely used benchmark problems from the UCI machine learning repository [51] and on a large data stream of intrusion detection data [40]. Note that the datasets from the UCI repository, despite their relatively small size and often lacking time tag, can be considered as pseudodata streams. For example, if credit card data are collected for a long period of time, an interesting data stream would be generated similar to the "credit card" dataset from [51]. The main reason these benchmark datasets were used in this paper is that a number of solutions exist in the literature, mostly based on offline classification approaches [35], and also on incremental versions [32]. The main aim of comparing *eClass* to existing classifiers is to illustrate that it can not only give similar or even better classification rates if working offline, but can also operate online on a per-sample basis, which is a computationally more efficient and flexible option. The robustness of *eClass* was also tested using noisy training data. Offline experiments were performed using tenfold cross-validation and

statistical averaging to compare the classification rate. Well-established tree-based classifiers C4.5 [52] and kNN [9] were used in the tests in the following two modes: 1) tenfold cross-validation and 2) an incremental version using a sliding window as described in [37]. *eClass* was run on the intrusion detection data online recursively. In what follows, each of the experimental datasets used will be briefly described (more details can be found in [40] and [51]) and the results tabulated and analyzed.

A. Benchmark Problems From UCI Repository

1) Sonar Data: This dataset contains patterns obtained from the reflected sonar signals from a metal cylinder at various angles and under various conditions. Based on the readings of these signals, it is required to classify the object as a rock or a mine (metal cylinder). Thus, there are two classes. The inputs consist of the reflected wave energy at 60 frequencies from different angles. This problem is a very realistic potential application of *eClass*. Similar experiments were performed at Lancaster University's Intelligent Systems Research Laboratory using the Pioneer 3DX mobile robot reported in [53].

2) Credit Card Dataset: This is a set of records about the information related to credit card application processing. The classifier should assist the decision to accept the application for a new credit card or not. This is a very realistic data stream scenario if we assume that a huge amount of data is required to be processed in a short time with new data possibly having a changing pattern. In the UCI repository, a "snapshot" of such a scenario is presented, but one can use this to make comparisons with offline classifiers, keeping in mind that *eClass* is designed to work in the more realistic scenario where the data are presented in real time and in significantly greater quantity. In this classification problem, there are 15 features describing the applicants, such as their annual income, age, marital status, etc. The number of classes is two: unsuccessful and successful applications.

3) Ten-Digit Dataset: This dataset contains the handwriting information from a pressure-sensitive tablet PC. The aim is to classify the ten digits from 0 to 9. In this way, the number of classes is ten. Sixteen features represent the readings from different areas of the tablet.

4) Ionosphere Dataset: This set contains 34 attributes of the pulse return from a radar that scans the Earth's ionosphere [51]. Based on these features, the classifier is required to judge whether the radar being tested is working or not. The features are obtained by analyzing the complex electromagnetic signal.

5) Pima Dataset: This is a database collected from the population of Phoenix, USA, which contains information about patients who show signs of diabetes [51]. The aim is to aid the diagnostic process by classifying the patients into "possible" and "no" diabetes classes using eight different features of the individual, such as blood pressure, number of times pregnant, age, etc. A real-life scenario could include a much larger database of a similar type, which is completed every day, and this would constitute a data stream. One can apply *eClass* to such a realistic scenario and the classifier will evolve, adapting to the new data. In this experiment, the main aim was to compare the performance to established classifiers (kNN and C4.5).

6) Wine Dataset: This dataset contains the results of a chemical analysis of wines grown in the same region of Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines [51].

For each of the aforementioned six datasets, a tenfold cross-validation was applied using *eClass*, kNN, and C4.5. This was repeated ten times with randomized sample order each time, and the results were averaged to eliminate the effect of order-dependency. As a result, 100 runs were performed for each approach. The results are listed in Table I. Note that *eClass* is one-pass (processes each sample just once at the time of its arrival and disregards it afterward), while both kNN and C4.5 use *all* training data (90% of the whole dataset) in the memory and make many iterations. The validation is done in all approaches on the remaining unused 10% of the data. The time required to process *one sample* is also presented in microseconds. Note that for the kNN and C4.5, this is derived as the time required to train the classifier *divided* by the number of training samples, while for *eClass*, this includes both training and prediction steps since *eClass* does not require pretraining and starts "from scratch." The overall training time for kNN and C4.5 is a multiplication of the number of training samples by the time shown in Table I.

The result demonstrates that *eClass* (especially *eClass1*) can compete well with established offline approaches even though the learning is performed in a single pass and a significantly smaller memory is used. Additionally, the structure of *eClass* is open, which allows capturing new data patterns.

In another experiment, in order to examine the robustness of *eClass*, we added random normally distributed 5% noise to the training datasets within the same experimental setting. The results are tabulated in Table II. They are very close to the results achieved without noise, which demonstrates the robustness of *eClass* that is due to the very conservative partitioning, which naturally avoids outliers (see Fig. 2 where the red point does not have a chance to become a prototype because there will be other points with higher potential).

Additionally, the proposed classifiers have been compared to the incremental versions of the well-established classifiers C4.5 and kNN, as described in [35]. The original datasets were divided into tenfolds, and each time, two consecutive folds have been used for training with the prediction performed on the next fold. This incremental, but not sample-by-sample, approach leads to sliding window versions of C4.5 and kNN. Note that *eClass* in its original form works on a per-sample basis and does not need a sliding window. The results are tabulated in Table III.

The results clearly demonstrate the superiority of *eClass* over the incremental versions of kNN and C4.5.

B. Intrusion Detection Data Stream

The second experiment was carried out on a real data stream called "intrusion detection dataset," which has been used in the Knowledge Discovery and Data Mining (KDD) 1999 Cup competition [40]. In this scenario, *eClass* works as an evolving

TABLE I
RESULTS OF THE TENFOLD CROSS-VALIDATION OFFLINE EXPERIMENT

		Sonar	Credit Card	PenDigit	Ionosphere	Pima	Wine	Average, %
eClass0	Classification rate %	60.76	80.57	83.14	74.6	69.38	92.44	**78.81**
	Standard deviation	13.11	7.89	2.24	9.1	6.22	7.09	
	Number of rules	9.21	8.47	39.78	6.75	6.31	9.94	
	Time, μs per sample	57	50	618	43	40	53	
eClass1	Classification rate %	76.57	85.95	96.83	87.71	76.74	97.22	**87.23**
	Standard deviation	9.52	4.26	2.02	5.91	4.75	4.36	
	Number of rules	6.45	8.69	16.26	4.99	6.07	6.4	
	Time, μs per sample	127	103	935	107	82	96	
C45	Classification rate %	74.31	85.3	89.5	90.71	73.73	92.13	**85.52**
	Standard deviation	8.9	4.03	9.23	4.9	4.4	6.3	
	Number of rules	7.22	7.48	2.36	7.45	7.48	4.6	
	Time, μs per sample	39	44	843	46	46	52	
kNN	Classification rate %	80.60	84.28	98.91	86.60	74.45	96.94	**86.91**
	Standard deviation	11.54	3.96	0.57	5.50	4.28	3.89	
	k	3	3	3	3	3	3	
	Time, μs per sample	30	50	769	40	45	28	

TABLE II
eClass WITH NOISY DATA (TEST FOR ROBUSTNESS)

	eClass0 (+5% noise)				*eClass1 (+5% noise)*			
	Rate	**std**	**rule**	**Time, μs per sample**	**Rate**	**std**	**rule**	**Time, μs per sample**
Sonar	60.32	14.69	9.71	55	76.52	9.5	5.38	130
Credit Card	79.35	642	8.08	52	83.78	3.28	9.07	100
PenDigit	81.21	2.28	37.63	627	96.21	2.23	16.79	928
Ionosphere	72.65	8.71	7.28	43	86.54	5.4	5.02	110
Pima	67.23	6	6.22	39	72.45	4.92	5.9	85
Wine	90.38	7.55	10.64	55	97.22	4.36	6.4	92
AVERAGE	**75.19**				**85.45**			

intrusion detector to classify between a normal connection to a server and different types of attacks (intrusion connections). The input data flow contains the details of the connections, such as protocol type, connection duration, login type, etc. In total, 40 features have been used [40]. Each connection can be categorized into five main classes (one normal class and four main intrusion classes: probe, DOS, U2R, R2L) [40]. There are 22 different types of attacks that were grouped into the four main types listed before (probe, DOS, U2R, R2L). The experimental setting is the same as the one used in the KDD 1999 Cup [40], taking 10% of the whole real raw data stream (over 5 000 000 data samples) for training and 311 029 data samples for testing. This was done to make a comparison with the results published for this data stream, including the results from the competition.

The results of the comparison of *eClass* with the winner of the competition and with the established approaches (kNN and C4.5) are tabulated in Table IV. Note that all the participants used *offline* approaches; some used expert knowledge [54], while *eClass* starts with no predefined rules and is fully automatic. The overall cost of the result (taking into account the importance and weight of each misclassification) calculated in the same manner as in [54] is 0.2480 for *eClass1* and 0.3020 for *eClass0*. Both results are better than the average result achieved by all the 24 participants in the competition (a cost of 0.3114). The result achieved by *eClass1* is practically the same as the one achieved by C4.5 and marginally worse than the result achieved by the winner [37]. Both C4.5 and the approach used by the winner (using bagging and boosting plus intensive preprocessing) are offline and require a batch set of training data. Note that *eClass1* achieved a better result on the rarest type of attack (R2L) than the winner of the competition. This is due to the ability of *eClass1* to adapt quickly to new data patterns.

The final test was performed on a small subset of the intrusion data (containing 4000 data samples randomly selected from the

TABLE III
ONLINE EXPERIMENT WITH UCI-BASED DATASETS

		Sonar	Credit Card	PenDigit	Ionosphere	Pima	Wine	Average, %
eClass0	rate %	62.16	78.18	83.12	72.42	69.43	87.87	**75.53**
	Std	2.33	9.47	1.43	6.53	2.76	3.82	
	Rule	10	14.6	40	6.3	6.3	7.17	
	Time, μs per sample	61	72	621	40	40	36	
eClass1	rate %	74.90	84.62	96.00	83.42	74.71	91.97	**84.27**
	Std	0.67	1.47	2.55	1.80	4.30	2.00	
	Rule	7	9.8	15.7	5	5.36	6.7	
	Time, μs per sample	136	112	889	108	70	93	
Incremental C45	rate %	62.35	70.20	85.55	83.88	59.8	91.19	**75.5**
	Std	7.3	6.37	8.99	5.2	5.7	6.5	
	Rule	6.89	6.45	2.19	7.37	5.93	4.3	
	Time, μs per sample	37	40	789	46	39	49	
Incremental kNN	rate %	72.50	83.17	97.69	81.07	70.39	95.59	**83.40**
	Std	17.73	6.04	0.47	9.16	4.22	5.21	
	K	3	3	3	3	3	3	
	Time, μs per sample	28	45	628	31	43	20	

TABLE IV
COMPARISON OF THE RESULTS FOR THE INTRUDER DETECTION PROBLEM

Method	normal	prob	DOS	U2R	R2L	Cost	Time, ms per sample
The winner	99.50%	83.30%	97.10%	13.20%	8.40%	**0.2331**	--
eClass 0	95.66%	76.31%	92.49%	20.61%	5.72%	**0.3020**	**3.90**
eClass 1	99.14%	63.13%	96.53%	11.84%	9.23%	**0.2480**	**6.98**
kNN [40]	99.60%	75.00%	97.30%	35.00%	0.60%	**0.2523**	--
Incremental kNN (k=3)	94.28%	72.61%	91.38%	25.00%	1.79%	**0.3366**	**6.52**
C4.5 [55]	97.89%	93.52%	97.08%	14.47%	1.21%	**0.2478**	--
Incremental C4.5	94.60%	51.54%	89.89%	14.94%	3.29%	**0.3726**	**8.39**

four subclasses met in the dataset) in order to test the ability of *eClass1* to respond quickly to new patterns by evolving the FRB while preserving the stability of the learning. *eClass1* starts to evolve from *an empty* rule base. The evolution of the classification rate is illustrated in Fig. 5. It has been achieved by testing the evolved classifier at different stages of learning. One can see the steady increase in the classification rate with the increase of the number of data samples met. The figure also demonstrates that *eClass1* is able to evolve five fuzzy rules automatically, reaching classification rates well over 90% after less than 500 samples. With a closer look [Fig. 5(b)], one can see that the curve is not ideally smooth—the drops in classification rate occur when a new type of attack is met. This ability to rapidly evolve and adapt is critical to detecting new types of attacks that may be launched in a real-life scenario. Therefore, we assess *eClass* as a better classifier overall, combining the ability to work on a per-sample basis, to quickly achieve high classification rates and to start "from scratch," while at the same time achieving results comparable to the well-known offline classifiers.

V. CONCLUSION

This paper introduced a family of FRB classifiers with an evolving structure (fuzzy rules) called *eClass* that includes *eClass0* and *eClass1* (*eClass1* can have single or multiple outputs). Both classifiers are based on eTS-type fuzzy systems. *eClass0* uses class labels as outputs. The proposed classifiers can start learning "from scratch." They can also be used to evolve (adapt and further develop the FRB structure) an initial classifier *iniClass*, if such a classifier is available in an interactive scheme. Starting "from scratch" (which is equivalent to an empty *iniClass* FRB) is a feature that is unique for *eClass* classifiers. The architecture of *eClass1* differs significantly—it is based on regression over the features producing an estimate of the possibility that a data sample may belong to a certain class. It attains higher classification rates, which compare favorably with results from well-established offline and incremental classifiers. Learning for the *eClass* family is based on a computationally efficient, recursive procedure, and it is therefore applicable for online real-time applications. The proposed approach addresses the problem of classification of streaming data (video, speech, financial data, sensory inputs in robotic and advanced industrial and Internet applications, etc.). It has been successfully tested on a number of benchmark problems simulating pseudodata streams to produce a comparison with the established approaches. It has also been successfully tested on an intrusion detection data stream where it has demonstrated its advantages. The results demonstrate that flexible classifiers can be generated online from streaming data, achieving high classification rates. The proposed method has the potential to be used in interactive mode or as a part of a system of collaborative classifiers.

APPENDIX

A. Recursive Expression of the Potential Using Cosine Distance

Combining (8) and (9) we get

$$P_k(z_k)=\frac{1}{1+\frac{1}{k-1}\sum_{i=1}^{k-1}\left[1-\frac{\sum_{j=1}^{n}z_k^j z_i^j}{\sqrt{\sum_{j=1}^{n}\left(z_k^j\right)^2\sum_{j=1}^{n}\left(z_i^j\right)^2}}\right]}. \tag{A1}$$

Reorganizing this, we get

$$P_k(z_k)=\frac{1}{2-\frac{1}{k-1}\frac{1}{\sqrt{\sum_{j=1}^{n}\left(z_k^j\right)^2}}\sum_{i=1}^{k-1}\frac{\sum_{i=1}^{n}z_k^j z_i^j}{\sqrt{\sum_{j=1}^{n}\left(z_i^j\right)^2}}}. \tag{A2}$$

Let us denote by A_k the following expression:

$$B_k=\sum_{i=1}^{k-1}\frac{\sum_{i=1}^{n}z_k^j z_i^j}{\sqrt{\sum_{i=1}^{n}\left(z_i^j\right)^2}}. \tag{A3}$$

B_k can be further simplified into

$$\begin{aligned}B_k&=\sum_{i=1}^{k-1}\frac{\sum_{j=1}^{n}z_k^j z_i^j}{\sqrt{\sum_{j=1}^{n}\left(z_i^j\right)^2}}\\&=\sum_{i=1}^{k-1}\frac{1}{\sqrt{\sum_{j=1}^{n}\left(z_i^j\right)^2}}\sum_{j=1}^{n}z_k^j\sqrt{\left(z_i^j\right)^2}\end{aligned} \tag{A4}$$

and finally into

$$B_k=\sum_{j=1}^{n}z_k^j\sum_{i=1}^{k-1}\sqrt{\frac{\left(z_i^j\right)^2}{\sum_{l=1}^{n}\left(z_i^l\right)^2}}. \tag{A5}$$

Let us further denote by b_k^j

$$b_k^j=\sum_{i=1}^{k-1}\sqrt{\frac{\left(z_i^j\right)^2}{\sum_{l=1}^{n}\left(z_i^l\right)^2}}. \tag{A6}$$

Then, we have

$$\begin{aligned}&B_k=\sum_{j=1}^{n}z_k^j b_k^j,\quad b_k^j=b_{k-1}^j+\sqrt{\frac{\left(z_k^j\right)^2}{\sum_{l=1}^{n}\left(z_k^l\right)^2}},\\&b_1^j=\sqrt{\frac{\left(z_1^j\right)^2}{\sum_{l=1}^{n}\left(z_1^l\right)^2}}.\end{aligned} \tag{A7}$$

From (A2), (A3), and (A7), we finally get (11).

Update of the Potential of the Focal Points (Prototypes): By definition (9), we have the *global* potential for each center as

$$P_k(z^*)=\frac{1}{1+\left(\sum_{i=1}^{k}d_{\cos}(z^*,z_i)\right)\Big/(k-1)} \tag{A8}$$

or if calculated based on $(k-1)$ data samples

$$P_{k-1}(z^*)=\frac{k-2}{k-2+\sum_{i=1}^{k-1}d_{\cos}(z_i,z^*)}. \tag{A8a}$$

Reorganizing (A8) and (A8a), we get

$$\begin{aligned}&P_k(z^*)\\&=\frac{1}{1+\left(\sum_{i=1}^{k-1}d_{\cos}(z_i,z^*)+d_{\cos}(z_k,z^*)\right)\Big/(k-1)}\end{aligned} \tag{A9}$$

$$\times\sum_{i-1}^{k-1}d_{\cos}(z_i,z^*)=(k-2)\left(\frac{1}{P_{k-1}(z^*)}-1\right). \tag{A10}$$

From (A9) and (A10), we finally get (12a).

In a similar way, if we start from the definition of local potential (10) and apply the same logic, we get (A9a) and (A10a), which when combined, give (12):

$$\begin{aligned}&P_k(z^*)\\&=\frac{1}{1+\left(\sum_{i=1}^{Q_k^l-1}d_{\cos}(z_i,z^*)+d_{\cos}(z_k,z^*)\right)/(Q_k^l-1)}\end{aligned} \tag{A9a}$$

$$\times\sum_{i-1}^{Q_k^l-1}d_{\cos}(z_i,z^*)=\left(Q_k^l-2\right)\left(\frac{1}{P_{k-1}(z^*)}-1\right). \tag{A10a}$$

B. Pseudocode of the *eClass0* Algorithm

Algorithm 1 ***eClass0***

Begin ***eClass0***
  Initialize ***eClass0*** by the first data sample, $z_1=[x_1,L_1]$; $(P_1)_1\leftarrow 1$ (or by *iniClass* if it exists)
  DO for a data sample **WHILE** data stream ends
    Read feature vector of the data sample, x_k;
    Calculate the membership to each of the fuzzy sets by (4);
    Calculate the rule firing strength by (3);
    Classify the sample using the winning class label, $\hat{L}_k$ by (2)
    At the **next time step** ($k\leftarrow k+1$) or whenever available after that
    Get the true label L_k of the sample;
    Calculate its potential, $P_k(z_k)$ using (11);
    Update potentials of the existing prototypes by (12);
    IF (13) holds
    THEN
      Add a new cluster center based at the new data point (14);
      initiate its potential to *1* as shown in (15);
      update spreads of membership functions of the rules in the sub-rule-base of the corresponding class by (17) or (18);
      IF (16) holds
      THEN Remove the rules for which it holds;
      END (IF)
    ELSE
      Ignore (do not change the cluster structure);
      Update spreads of membership functions of the rules in the sub-rule-base of the corresponding class by (17) or (18).
      Optional: Remove rules for which (25) holds.
    END (IF THEN ELSE)
  END (DO…WHILE)
END (*eClass0*)

C. Pseudocode of the Algorithm eClass1

Algorithm 2 ***eClass1***

Begin ***eClass1***

Initialize ***eClass1*** by the first data sample, $z_1=[x_1,L_1]$; $(P_1)_1 \leftarrow 1$ (or by *iniClass* if it exists)

DO for a data sample **WHILE** data stream ends

Read feature vector of the data sample, x_k;

Calculate the membership to each of the fuzzy sets by (4);

Calculate the rule firing strength by (3);

Determine the winning class label, $\hat{L}_k$ by (22) for *eClass1-MISO* or by (23) for *eClass1-MIMO*;

At the **next time step** ($k \leftarrow k+1$) or whenever available after that

Get the true class label, L_k;

IF (use *MIMO* version and the class label have not been previously seen)

THEN (initialize the potential by *1*: $(P_k)_k \leftarrow 1$, augment the consequent parameters matrix by adding a column with $(n+1)$ zeros: $\Theta_k^i = \left[\Theta_k^i \quad 0_{(n+1)x1}\right]$ and $K \leftarrow K+1$)

ELSE (Calculate its potential, $P_k(z_k)$, using (11) but applied to the **global** potential);

Update potentials of the existing prototypes by (12a);

IF (13) holds

THEN

Add a new cluster center based at the new data point (14);

initiate its **global** potential to *1* as shown in (15);

update spreads of membership functions by (17) or (18);

IF (16) holds

THEN Remove the rules for which it holds;

END (IF)

ELSE

Ignore (do not change the cluster structure);

Update spreads of membership functions by (17) or (18);

Optional: Remove rules for which (25) holds.

END (IF THEN ELSE)

Update the consequents parameters by (26)-(27).

END (DO…WHILE)

END (*eClass1*)

ACKNOWLEDGMENT

The authors would like to thank Dr. E. Lughofer, J. Kepler University, Linz, Austria, and Dr. D. Filev, Ford Motor Company, for their comments and useful discussions, and Mr. M. Everett for proofreading the manuscript.

REFERENCES

[1] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if–then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 260–270, Aug. 1995.

[2] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, "Ten years of genetic fuzzy systems: Current framework and new trends," *Fuzzy Sets Syst.*, vol. 141, no. 1, pp. 5–31, 2004.

[3] L. Kuncheva, *Fuzzy Classifiers*. Heidelberg, Germany: Physica-Verlag, 2000.

[4] L. I. Kuncheva, "How good are fuzzy if–then classifiers?" *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 4, pp. 501–509, Aug. 2000.

[5] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," *Fuzzy Sets Syst.*, vol. 89, pp. 277–288, 1997.

[6] J. C. Bezdek, J. Keller, R. Krishnapuram, and N. R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Norwell, MA: Kluwer, 1999.

[7] H. Ishibuchi, T. Nakashima, and M. Nii, *Classification and Modeling With Linguistic Granules: Advanced Information Processing*. Berlin, Germany: Springer-Verlag, 2004.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Heidelberg, Germany: Springer-Verlag, 2001.

[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Chichester, U.K.: Wiley–Interscience, 2000.

[10] K. M. A. Chai, H. T. Ng, and H. L. Chieu, "Bayesian online classifiers for text classification and filtering," in *Proc. SIGIR 2002*, Tampere, Finland, Aug. 11–15, pp. 97–104.

[11] R. Jin and G. Agrawal, "Efficient decision tree construction on streaming data," in *Proc. ACM SIGKDD*, 2003, pp. 571–576.

[12] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees*. Boca Raton, FL: Chapman & Hall, 1993.

[13] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[14] V. N. Vapnik, *The Statistical Learning Theory*. New York: Springer-Verlag, 1998.

[15] L.-X. Wang, "Fuzzy systems are universal approximators," in *Proc. 1st IEEE Conf. Fuzzy Syst. (FUZZ-IEEE)*, San Diego, CA, Mar. 8–12, 1992, pp. 1163–1170.

[16] F. Hopner and F. Klawonn, "Obtaining interpretable fuzzy models from fuzzy clustering and fuzzy regression," in *Proc. 4th Int. Conf. Knowl.-Based Intell. Eng. Syst. (KES)*, Brighton, U.K., 2000, pp. 162–165.

[17] J. G. Marin-Blazquez and Q. Shen, "From approximative to descriptive fuzzy classifiers," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 4, pp. 484–497, Aug. 2002.

[18] J. Casillas, O. Cordon, and F. Herrera, "COR: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 4, pp. 526–537, Aug. 2002.

[19] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Jan. 1985.

[20] P. Domingos and G. Hulten, "Catching up with the data: Research issues in mining data streams," presented at the Workshop Res. Issues Data Mining Knowl. Discovery, Santa Barbara, CA, 2001.

[21] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *From Data Mining to Knowledge Discovery: An Overview, Advances in Knowledge Discovery and Data Mining*. Cambridge, MA: MIT Press, 1996.

[22] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[23] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[24] A. S. Hornby, *Oxford Advance Learner's Dictionary*. London, U.K.: Oxford Univ. Press, 1974.

[25] J. Gomez, F. Gonzalez, D. Dasgupta, and O. Nasaroui, "Complete expression tree for evolving fuzzy classifier systems with genetic algorithms," in *Proc. North Amer. Fuzzy Inf. Process. Soc. Conf. Fuzzy Logic Internet (NAFIPS-FLINT)*, 2002, pp. 469–474.

[26] P. Angelov, *Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems*. Berlin, Germany: Springer-Verlag, 2002.

[27] P. Angelov and D. Filev, "An approach to on-line identification of evolving Takagi–Sugeno models," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 484–498, Feb. 2004.

[28] P. Angelov and R. Buswell, "Identification of evolving rule-based models," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 5, pp. 667–677, Oct. 2002.

[29] P. Angelov, V. Giglio, C. Guardiola, E. Lughofer, and J. M. Lujan, "An approach to model-based fault detection in industrial measurement systems with application to engine test benches," *Meas. Sci. Technol.*, vol. 17, no. 7, pp. 1809–1818, 2006.

[30] J. Macias, P. Angelov, and X. Zhou, "Predicting quality of the crude oil distillation using evolving Takagi–Sugeno fuzzy models," in *Proc. 2006 Int. Symp. Evol. Fuzzy Syst.*, pp. 201–207.

[31] N. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.
[32] F. Ferrer-Troyano, J. S. Aguilar-Ruiz, and J. C. Riquelme, "Incremental rule learning based on example nearness from numerical data," in *Proc. 2005 ACM Symp. Appl. Comput.*, 2008, pp. 568–572.
[33] S. Pang, S. Ozawa, and N. Kasabov, "Incremental linear discriminant analysis for classification of data streams," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 35, no. 5, pp. 905–914, Oct. 2005.
[34] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–712, Sep. 1992.
[35] H. Alhammady and K. Ramamohanarao, "Mining emerging patterns and classification in data streams," in *Proc. 2005 IEEE/WIC/ACM Int. Conf. Web Intell. (WI 2005),* Sep., pp. 272–275.
[36] L. Golab and M. T. Ozsu, "Issues in data stream management," in *Proc. ACM SIGMOD Conf.*, Jun. 2003, vol. 32, no. 2, pp. 5–14.
[37] B. Pfahringer, "Wining the KDD99 Classification Cup: Bagged boosting," in *Proc. ACM SIGKDD*, Jan. 2000, vol. 1, no. 2, pp. 65–66.
[38] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learning*, vol. 23, no. 1, pp. 69–101, 1996.
[39] R. Klinkenberg and T. Joachims, "Detection concept drift with support vector machines," in *Proc. 7th Int. Conf. Mach. Learning (ICML)*, 2000, pp. 487–494.
[40] C. Elkan. (2007, Jan. 27). Results of the KDD'99 Knowledge Discovery Contest [Online]. Available: http://www-cse.ucsd.edu/users/elkan/clresults.html
[41] J. Roubos, M. Setnes, and J. Abonyi, "Learning fuzzy classification rules from data," *Inf. Sci.*, vol. 150, pp. 77–93, 2003.
[42] F. Klawonn and P. E. Klement, "Mathematical analysis of fuzzy classifiers," *Lect. Notes Comput. Sci.*, vol. 1280, pp. 359–370, 1997.
[43] P. Angelov and D. Filev, "Simpl_eTS: A simplified method for learning evolving Takagi–Sugeno fuzzy models," in *Proc. 2005 IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Reno, NV, May 22–25, pp. 1068–1073.
[44] P. Angelov, "An approach for fuzzy rule-base adaptation using on-line clustering," *Int. J. Approx. Reason.*, vol. 35, no. 3, pp. 275–289, Mar. 2004.
[45] K. J. Astroem and B. Wittenmark, *Adaptive Control*. Reading, MA: Addison-Wesley, 1989.
[46] R. R. Yager and D. P. Filev, "Learning of fuzzy rules by mountain clustering," in *Proc. SPIE Conf. Appl. Fuzzy Logic Technol.*, Boston, MA, 1993, pp. 246–254.
[47] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.
[48] P. Angelov and X. Zhou, "Evolving fuzzy systems from data streams in real-time," in *Proc. 2006 Int. Symp. Evol. Fuzzy Syst.*, Sep. 7–9, pp. 29–35.
[49] P. Angelov, C. Xydeas, and D. Filev, "On-line identification of MIMO evolving Takagi–Sugeno fuzzy models," in *Proc. Int. Joint Conf. Neural Netw. Int. Conf. Fuzzy Syst. (IJCNN-FUZZ-IEEE)*, Budapest, Hungary, Jul. 25–29, 2004, pp. 55–60.
[50] G. Klir and T. Folger, *Fuzzy Sets, Uncertainty and Information*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[51] A. Asuncion and D. J. Newman. (2007). UCI Machine Learning Repository [Online]. School of Information and Computer Science, University of California, Irvine. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[52] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *J. AI Res.*, vol. 4, pp. 77–90, 1996.
[53] X. Zhou and P. Angelov, "An approach to autonomous self-localization of a mobile robot in completely unknown environment using evolving fuzzy rule-based classifier," in *Proc. 2007 IEEE Int. Symp. Comput. Intell. Appl. Defense Security Symp. Series Comput. Intell. (SSCI 2007)*, Honolulu, HI, Apr. 1–5, pp. 131–138 (ISBN 1-4244-0698-6).
[54] V. Mikheev, A. Vopilov, and I. Shabalin, "The MP13 approach to the KDD'99 classifier learning contest," *ACM SIGKDD Explorations Newslett.*, vol. 1, no. 2, pp. 76–77, 1999.
[55] A. D. Joshi. (2004). Applying the wrapper approach for auto discovery of under-sampling and over-sampling percentages on skewed datasets, M.Sc. Thesis, Univ. South Florida, Tampa [Online]. pp. 1–77. Available: http://etd.fcla.edu/SF/SFE0000491/Thesis-AjayJoshi.pdf
[56] L. Zadeh, "Fuzzy sets as the basis for a theory of possibility," *Fuzzy Sets Syst.*, vol. 1, pp. 3–28, 1978.

Plamen P. Angelov (M'99–SM'04) received the M.Eng. degree in electronics and automation from Sofia Technical University, Sofia, Bulgaria, in 1989 and the Ph.D. degree in optimal control from Bulgaria Academy of Sciences, Sofia, Bulgaria, in 1993.

He spent over ten years as a Research Fellow working on computational intelligence and control. During 1995–1996, he was at Hans-Knoell Institute, Jena, Germany. In 1997, he was a Visiting Researcher at the Catholic University, Leuvain-la-neuve, Belgium. In 2007, he was a Visiting Professor at the University of Wolfenbuettel-Braunschweig, Germany. He is currently a Senior Lecturer (Associate Professor) at Lancaster University, Lancaster, U.K. He has authored or coauthored over 100 peer-reviewed publications, including the book *Evolving Rule Based Models: A Tool for Design of Flexible Adaptive Systems* (Springer-Verlag, 2002), and over 30 journal papers, and is a holder of a patent (2006). He is a Member of the Editorial Boards of three international scientific journals. He also holds a portfolio of research projects including a participation in a £32M program ASTRAEA. He has received funding from the industry, European Union (EU), U.S., U.K. Research Council, etc. His current research interests include adaptive and evolving (self-organizing) fuzzy systems as a framework of an evolving computational intelligence. He is also engaged in online and real-time identification and design of such systems with evolving (self-developing) structure, intelligent data processing, particularly in evolving fuzzy-rule-based models, self-organizing and autonomous systems, and intelligent (inferential) sensors.

Dr. Angelov is a member of the Technical Committee (TC) on Fuzzy Systems, Vice Chair of the TC on Standards, Computational Intelligence Society, Chair of the Task Force on Adaptive Fuzzy Systems, and member of the Autonomous Systems Working Group of the North-West Science Council of U.K. He serves regularly on the technical/program committees of leading international conferences on different aspects of computational intelligence.

Xiaowei Zhou (S'06) received the Bachelor's degree in computing science from Nanjing University of Science and Technology, Nanjing, China, in 2003, and the Master's degree (with distinction) in IT and data communications in 2004 from the Department of Communication Systems, Lancaster University, Lancaster, U.K., where he is currently working toward the Ph.D. degree at Infolab 21.

He has authored or coauthored over a dozen of peer-reviewed papers on evolving intelligent systems. His current research interests include evolving intelligent systems, robotics, image processing, and their applications under the framework of computational intelligence and machine learning.

Mr. Zhou was a recipient of the IEEE Student Travel Grant in 2006 to attend the World Congress on Computational Intelligence and a number of other smaller travel grants. He is the main author of the fuzzy car algorithm, which was ranked third in the competition during the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007), London, U.K., July.