Improving Premise Structure in Evolving Takagi-Sugeno Neuro-Fuzzy Classifiers

Abdullah Almaksour, Eric Anquetil

Noname manuscript No.
(will be inserted by the editor)

Improving Premise Structure in Evolving Takagi-Sugeno Neuro-Fuzzy Classifiers

Abdullah Almaksour · Eric Anquetil

the date of receipt and acceptance should be inserted later

Abstract We present in this paper a new method for the design of evolving neuro-fuzzy classifiers. The presented approach is based on a first-order Takagi-Sugeno neuro-fuzzy model. We propose a modification on the premise structure in this model and we provide the necessary learning formulas, with no problem-dependent parameters. We demonstrate by the experimental results the positive effect of this modification on the overall classification performance.

Keywords Incremental learning · Takagi-Sugeno · neuro-fuzzy

1 Introduction

Classification techniques appear frequently in many application areas, and become the basic tool for almost any pattern recognition task. The main problem in classification is to induce a classifier from a set of data samples. A large amount of samples is needed to set up and evaluate a classification system that can achieve high accuracy, and it is practically very difficult to have such number of samples covering the whole feature space. Therefore, life-long classifier adaptation becomes more and more an essential point. Moreover, in many application contexts, the classifier needs to take into account new unseen classes and to integrate them in the classification process, which increases the need for "evolving" classifiers.

One good example of such applications is the use of online handwriting gesture classifiers that aims at facilitating interactions with computers using pen-based interfaces like whiteboards, tablet PCs, PDA...Etc. The main drawback in the current existing systems is that they are trained "offline" on a specific group of gestures and then implemented to operate without changing their structure during the use. This fixed structure does not allow the user to choose his own set of gestures or to add new

Abdullah Almaksour · Eric Anquetil
INSA de Rennes, Avenue des Buttes de Coesmes, F-35043 Rennes
CNRS, UMR IRISA, Campus de Beaulieu, F-35042 Rennes
Université Européenne de Bretagne, France E-mail: {Abdullah.Almaksour, Eric.Anquetil}@irisa.fr

ones to assign them to new interactive commands (according to his special needs), for example.

In our work, we aim at building a handwriting classifier, on-the-fly, from scratch and using only few data. Thus, the classifier will be incrementally adapted to achieve high recognition rates as soon as possible and to keep the system robust when introducing new unseen classes at any moment in the life-long learning process.

In order to clarify the meaning of the terms "incremental" and "online" learning, one can compare them to the "batch" and "offline" learning as follows [Kasabov (2007)]:

- **Batch versus incremental learning:** In a batch mode of learning a predefined training dataset is learned by the system through propagating this dataset several times through the system. Each time the system optimizes its structure based on the average value of the goal function over the whole dataset. The incremental mode of learning is concerned with learning each data sample separately and the data might exist only for a short time. After introducing each data sample, the system makes changes in its structure to optimize the goal function. An incremental learning algorithm must learn new data without fully destroying the knowledge learned from old data and without the need to retrain the system on the old and the new data.
- **Offline versus online learning:** In an offline learning mode, the system is trained on a specific dataset and then implemented to operate in a real environment, without changing its structure during the use. While in an online learning mode, the model learns from new data during its use.

An incremental learning algorithm is defined in [Polikar et al. (2001)] as being one that meets the following criteria: it should be able to learn additional information from new data; it should not require access to the original data (i.e. data used to train the existing classifier); it should preserve previously acquired knowledge (it should not suffer from *catastrophic forgetting*, significant loss of original learned knowledge); and it should be able to accommodate new classes that may be introduced with new data. Many of the existing "incremental learning" algorithms are not truly incremental because at least one of the mentioned criteria is violated. These criteria can be briefly expressed by the so called "plasticity-stability dilemma"[Zwickel and Wills (2005)], which is that a system must be able to learn in order to adapt to a changing environment but that constant change can lead to an unstable system that can learn new information only by forgetting everything it has so far learned.

We can distinguish two main types of incremental learning algorithms: algorithms for parameter incremental learning and algorithms for structure incremental learning. The incremental learning of parameters can be considered as "adaptation" algorithm. The structure in such systems is fixed and initialized at the beginning of the learning process, and the system parameters are learned incrementally according to newly available data. Some examples of these systems are presented in [Jang (1993)] and [Mouchere et al. (2007)].

Most of structure incremental learning algorithms are based on the principle of the ART clustering algorithm [Carpenter and Grossberg (1988)], such as [Carpenter et al. (1992)], [Sadri et al. (2006)], [Gary G. (2001)]. The main problem of these systems is that

they are sensitive to the selection of the vigilance parameter, to noise level in the training data and to the order in which training data are presented.

In an online incremental learning algorithm, the training set is not available a priori, since the learning examples come over time. Although online learning systems can continuously update and improve their models, not all of them are necessarily based on a real incremental approach. For some systems the model is completely rebuilt at each step of learning using all available data, they are systems with "full instance memory" [Reinke and Michalski (1988)]. On the other hand, if the learning algorithm modifies the model using only the last available learning example, it is called a system with "no instance memory" [Littlestone (1991)]. A third category is that of systems with "partial instance memory", which select and maintain a reduced subset of learning examples to use them in the next learning step [Aha et al. (1991)].

Evolving neuro-fuzzy models [Angelov and Filev (2003)] have been successfully used in many modeling and classification problems during the last years. Several incremental learning algorithms have been proposed for the evolving structure identification and the recursive parameters learning (Denfis[Song and Kasabov (2000)], Flexfis[Lughofer (2008)], eTS[Angelov and Filev (2004)], eTS+[Angelov (2010)] etc.). One of the most used neuro-fuzzy models is the First-order Takagi-Sugeno fuzzy model. It usually has a linear consequent part and its premise part can be learned from data (no expert is needed). These models can address problems with either single output or multi-output. It consists of a set of fuzzy rules of the following form:

$$\textbf{Rule}_i : \textbf{IF } \mathbf{x} \textit{ is close to } P_i \textbf{THEN } y_i^1 = l_i^1(\mathbf{x}), ..., \; y_i^k = l_i^k(\mathbf{x}) \tag{1}$$

where $l_i^m(\mathbf{x})$ is the linear consequent function of the rule i for the class m. The Prototype P is defined by a center and a fuzzy zone of influence. A membership function must be defined in order to calculate the "closeness" degree between x and P (considering its center and its fuzzy zone of influence).

In the existing approaches, the zone of influence has often a hyper-spherical form [Yager and Filev (1993)] [Angelov (2004)]. It has then the same radius for all the feature space dimensions, and it can be represented by a diagonal covariance matrix with identical values on the diagonal.

More sophisticated form is presented in [Angelov and Zhou (2006)] that allows the zone radius to have different values for the different dimensions (still diagonal covariance matrix but with different values on the diagonal). This ability results in hyper-elliptical zones where the ellipses are still parallel to the feature space axes.

In this paper, we go a step ahead in the structure of the fuzzy prototypes in first order TS models by allowing them to have a hyper-elliptical form non-parallel to the feature space axes. This form enables the model to take into consideration the correlations that can exist between the features, and is represented by normal (non-diagonal) covariance matrices. We calculate the prototype centers and covariance matrices in a recursive way with zero problem-dependent parameters. We use a membership function based on a well-known multivariate probability distribution function. By experimenting this structure on our handwritten gesture problem and other benchmark multi-class problems, we show its useful effect on the overall system output and the linear consequent learning. We also propose two different methods for incremen-

tal adaptation of prototypes, and we compare experimentally these two incremental learning models with a well-known reference model.

The rest of the paper is organized as follows: We describe in Section 2 the architecture of our neuro-fuzzy classifier. Then, the different elements of the used online incremental learning algorithm are detailed in Section III. We present our experimental results in Section IV before concluding in Section 5.

2 System architecture

As aforementioned, our system is based on first-order Takagi-Sugeno (TS) fuzzy inference system. It consists of a set of fuzzy rules of the following form:

$$\textbf{Rule}_i : \textbf{IF } \mathbf{x} \textit{ is close to } P_i \textbf{THEN } y_i^1 = l_i^1(\mathbf{x}), ..., \ y_i^k = l_i^k(\mathbf{x}) \tag{2}$$

where $l_i^m(\mathbf{x})$ is the linear consequent function of the rule i for the class m:

$$l_i^m(\mathbf{x}) = \boldsymbol{\pi}_i^m \mathbf{x} = a_{i0}^m + a_{i1}^m x_1 + a_{i2}^m x_2 + ... + a_{in}^m x_n \tag{3}$$

where n is the size of the input vector. Singleton consequences ($l_i^m(\mathbf{x}) = a_i^m$) are sometimes used instead of linear functions in order to get simpler models that are called zero-order TS models. To find the class of $\mathbf{x}$, its membership degree $\beta_i(\mathbf{x})$ to each fuzzy prototype is first computed. After normalizing these membership degrees, the sum-product inference is used to compute the system output for each class:

$$y^m(\mathbf{x}) = \sum_{i=1}^{r} \bar{\beta}_i(\mathbf{x}) \ l_i^m(\mathbf{x}) \tag{4}$$

where r is the number of fuzzy rules in the system. The winning class label is given by finding the maximal output and taking the corresponding class label as response:

$$class(\mathbf{x}) = y = argmax \ y^m(\mathbf{x}) \qquad m = 1, .., k \tag{5}$$

The membership degree can be calculated in many ways. For hyper-spherical or axes-parallel hyper-elliptical prototypes, the membership degree can be computed depending on the prototype center $\boldsymbol{\mu}_\mathbf{i}$ and the radius value σ_i (the same value in all the dimensions for the former, and different values for the later). In this case, the Gaussian membership function is generally used. The value of $\beta_i(\mathbf{x})$ can then be computed as follows:

$$\beta_i(\mathbf{x}) = \prod_{j=1}^{n} \exp(-\frac{\|x - \mu_i\|_j^2}{2\sigma_{ij}^2}) \tag{6}$$

It must then be normalized as follows:

$$\bar{\beta}_i(\mathbf{x}) = \frac{\beta_i(\mathbf{x})}{\sum_{j=1}^{r} \beta_j(\mathbf{x})} \tag{7}$$

In our model, where the fuzzy influence zones of the prototype take a rotated hyper-elliptical form, the membership degree is computed by the prototype center $\boldsymbol{\mu_i}$ and its variance-covariance matrix A_i:

$$A_i = \begin{bmatrix} \sigma_1^2 & c_{12} & ... & c_{1n} \\ c_{21} & \sigma_2^2 & ... & c_{2n} \\ ... & ... & ... & ... \\ c_{n1} & c_{n2} & ... & \sigma_n^2 \end{bmatrix}_i \tag{8}$$

where $c_{12}(= c_{21})$ is the covariance of x_1 and x_2, and so on. In order to calculate the membership degree, we use the multivariate Cauchy probability distribution. The value of $\beta_i(\mathbf{x})$ is computed in our model as follows:

$$\beta_i(\mathbf{x}) = \frac{1}{2\pi\sqrt{|A_i|}} \left[1 + (\mathbf{x} - \boldsymbol{\mu_i})^t A_i^{-1} (\mathbf{x} - \boldsymbol{\mu_i})\right]^{-\frac{n+1}{2}} \tag{9}$$

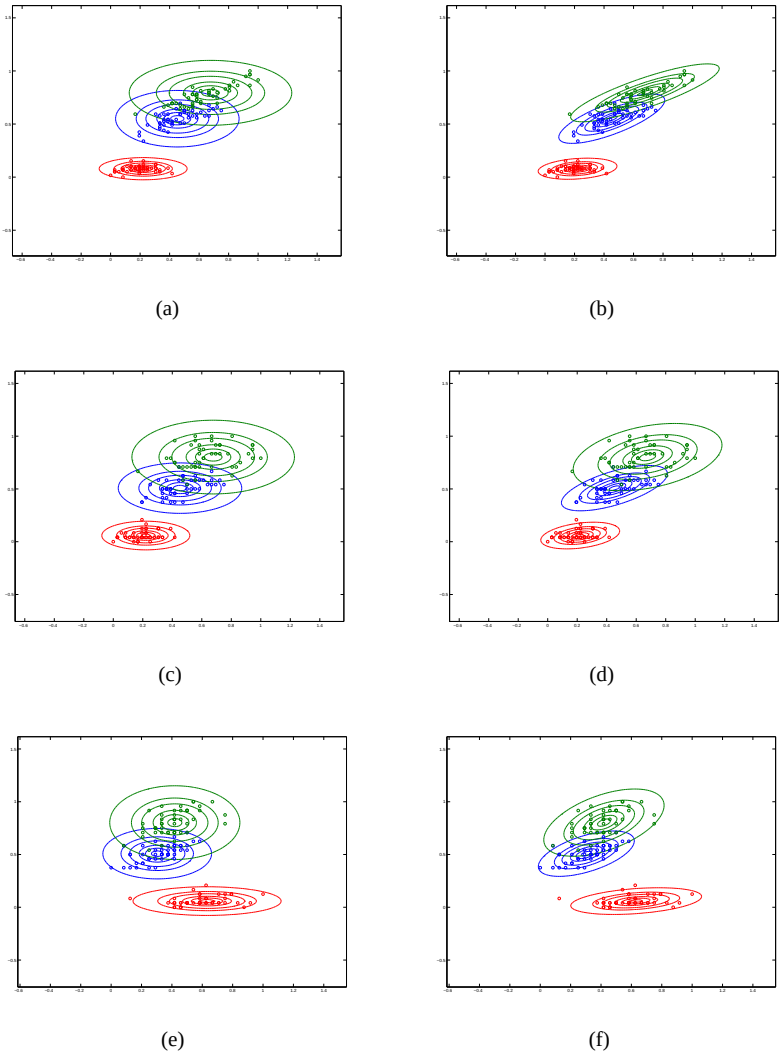
The difference between the two types of premises is illustrated in fig.1 (data from "Iris" dataset [Frank and Asuncion (2010)]).

3 On-line incremental learning algorithm

The incremental learning algorithm of our model consists of three different tasks: the creation of new rules, the adaptation of the existing rule's premises, and the tuning of the linear consequent parameters. These three tasks must be done in an online incremental mode and all the needed calculation must be completely recursive. For the incremental rule creation, we adapt a technique of incremental clustering. Coupling this technique with the incremental adaptation of the premises of the rules is part of the originality of our learning model. We propose two different methods for adapting the premises, and we use a modified version of the recursive least squares method for estimating the linear consequent parameters in incremental manner.We will describe in this section the different parts of the learning algorithm.

3.1 Incremental clustering

In an online incremental learning problem, training data become available continuously, and the system must be learned using the first-arrived data, and then continue to evolve in a transparent manner from the user viewpoint. Classic clustering algorithms need to know all the objects in order to perform the clustering and each time we modify the set of objects, it is necessary to cluster the whole collection again. As mentioned in the introduction, a very important criterion of an efficient incremental learning solution is avoiding (or minimizing) access to previous learning data. Thus, we need algorithm able to update the clusters each time a new data become available, without neither rebuilding the whole set of clusters, nor requiring access to previous data.



(a) (b)

(c) (d)

(e) (f)

Fig. 1 The zone of influence of a prototype is parallel to the axes in (a), (c) and (e), while the rotated zones in (b), (d) and (f) through the use a multidimensional probability distribution result in a more accurate data modeling.

When introducing a new training sample in an online learning mode, it will either reinforce the information contained in the previous data and represented by the current clustering, or bring enough information to form a new cluster or modify an existing one. The importance of a given sample in the clustering process can be evaluated by its *potential* value. The potential of a sample is defined as inverse of the sum of dis-

tances between a data sample and all the other data samples [Yager and Filev (1993)]:

$$Pot(\mathbf{x}(t)) = \frac{1}{1 + \sum_{i=1}^{t-1} \|x(t) - x(i)\|^2} \tag{10}$$

A recursive method for the calculation of the potential of a new sample was introduced in [Angelov and Filev (2004)], which made this technique a promised solution for any incremental clustering problem. The recursive formula avoids memorizing the whole previous data but keeps - using few variables - the density distribution in the feature space based on the previous data:

$$Pot(\mathbf{x}(t)) = \frac{t-1}{(t-1)\alpha(t) + \gamma(t) - 2\zeta(t) + t - 1} \tag{11}$$

where

$$\alpha(t) = \sum_{j=1}^{n} x_j^2(t) \tag{12}$$

$$\gamma(t) = \gamma(t-1) + \alpha(t-1), \quad \gamma(1) = 0 \tag{13}$$

$$\zeta(t) = \sum_{j=1}^{n} x_j(t)\eta_j(t), \quad \eta_j(t) = \eta_j(t-1) + x_j(t-1), \quad \eta_j(1) = 0 \tag{14}$$

Introducing a new sample affects the potential values of the centers of the existing clusters, which can be recursively updated by the following equation:

$$Pot(\mu_i) = \frac{(t-1)Pot(\mu_i)}{t - 2 + Pot(\mu_i) + Pot(\mu_i)\sum_{j=1}^{n} \|\mu_i - x(t-1)\|_j^2} \tag{15}$$

If the potential of the new sample is higher than the potential of the existing centers then this sample will be a center of a new cluster and a new fuzzy rule will be formed in the case of our neuro-fuzzy model. So, the center of the new prototype $\boldsymbol{\mu}_{r+1} = \mathbf{x}_k$ and its covariance matrix $A_{r+1} = \epsilon I$, where I is the identity matrix of size n and ϵ is a problem-independent parameter and can generally be set to 10^{-2}.

3.2 Premise adaptation

This adaptation process allows to incrementally update the prototype centers coordinations according to each new available learning data, and to recursively compute the prototype covariance matrices in order to give them the rotated hyper-elliptical form. For each new sample $\mathbf{x}_k$, the center and the covariance matrix of the prototype that has the highest activation degree are updated. We present two different approaches of adaptation; the first method is purely statistical, while the second takes into account the error or the confusion during the recognition of the new example.

3.2.1 Statistical recursive adaptation

In this method, the center coordination of the selected prototype is recalculated as follows:

$$\boldsymbol{\mu}_i = \frac{s_i - 1}{s_i}\boldsymbol{\mu}_i + \frac{1}{s_i}\mathbf{x}_t \tag{16}$$

where s_i represents the number of updates that have been applied so far on this prototype. The covariance matrix is recursively computed as follows:

$$A_i = \frac{s_i - 1}{s_i}A_i + \frac{1}{s_i - 1}(\mathbf{x}_t - \boldsymbol{\mu}_i)(\mathbf{x}_t - \boldsymbol{\mu}_i)^T \tag{17}$$

For practical issues, since the membership degree can be calculated using only $A^{-1}(|A| = \frac{1}{|A^{-1}|})$, and in order to avoid any matrix inversion, we use an updating rule for A^{-1} directly [De Backer and Scheunders (2001)]:

$$A_i^{-1} = \frac{A_i^{-1}}{1-\alpha} - \frac{\alpha}{1-\alpha} \cdot \frac{(A_i^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_i)) \cdot (A_i^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_i))^T}{1 + \alpha((\mathbf{x}_t - \boldsymbol{\mu}_i)^T A_i^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_i))} \tag{18}$$

where $a = \frac{1}{s_i - 1}$.

3.2.2 Confusion-driven recursive adaptation

We propose a second method of adaptation in which the adjustment is made in "supervised" manner. To guide the adaptation of premises to improve system performance, we focus on examples that are either misrecognized, or "hardly" well recognized. This is translated by adding in the adaptation formulas a weight w that will be calculated for each new instance and it represents the importance to be assigned to this example in the process of adaption. The adaptation formulas are then given as follows:

$$\boldsymbol{\mu}_i = \frac{s_i - w}{s_i}\boldsymbol{\mu}_i + \frac{w}{s_i}\mathbf{x}_t \tag{19}$$

$$A_i = \frac{s_i - w}{s_i}A_i + \frac{w}{s_i - 1}(\mathbf{x}_t - \boldsymbol{\mu}_i)(\mathbf{x}_t - \boldsymbol{\mu}_i)^T \tag{20}$$

$$w = 1 - [y^c - y^{nc}] \quad w \in [0, 2] \tag{21}$$

where

$$y^{nc} \; = \; argmax \; y^m \qquad m = 1..k \;\; \& \;\; m \neq c \tag{22}$$

In words, w is inversely proportional to the difference between the true label score of $\mathbf{x}_t$ (y^c) and the highest value between the other (wrong) classes' scores (y^{nc}). Thus, the value of w tends to 0 when $\mathbf{x}_t$ is "strongly" well recognized, and to 2 when it is misrecognized.

3.3 Linear consequent tuning

The tuning of the linear consequent parameters in a first-order TS model can be done using weighted Recursive Least Square method (wRLS) [Angelov et al. (2008)]. The wRLS method can be applied in either local or global manner. Consequent parameters of each rule are optimized separately in the former, while parameters of all the rules are optimized together in the latter. Let Π be the matrix of all the linear consequents parameters in first-order FIS :

$$\Pi = \begin{bmatrix} \boldsymbol{\pi}_1^1 & \boldsymbol{\pi}_1^2 & ... & \boldsymbol{\pi}_1^k \\ \boldsymbol{\pi}_2^1 & \boldsymbol{\pi}_2^2 & ... & \boldsymbol{\pi}_2^k \\ ... & ... & ... & ... \\ \boldsymbol{\pi}_r^1 & \boldsymbol{\pi}_r^2 & ... & \boldsymbol{\pi}_r^k \end{bmatrix} \tag{23}$$

where m is the number of classes, and r is the number of fuzzy rules; It can be globally and incrementally estimated by:

$$\Pi_t = \Pi_{t-1} + C_t \psi_t (Y_t - \psi_t^T \Pi_{t-1}) \; ; \quad \Pi_1 = 0 \tag{24}$$

$$C_t = C_{t-1} - \frac{C_{t-1}\psi_t \psi_t^T C_{t-1}}{1 + \psi_t^T C_{t-1} \psi_t} \; ; \quad C_1 = \Omega I \tag{25}$$

where $\psi_t = [\beta_1(\mathbf{x}_t)\mathbf{x}_t, \beta_2(\mathbf{x}_t)\mathbf{x}_t, ..., \beta_r(\mathbf{x}_t)\mathbf{x}_t]$ is the input vector weighted by the activation levels of the prototypes, I is the identity matrix and Ω is a large positive number, typically between 100 and 10,000. Small values of Ω slow down the learning, while too large Ω can prevent parameters converging properly. The value must then be estimated to be a good compromise between these two points. Ω = 1000 is adequate for most cases. Y_t is a 1-by-k target vector of the input vector x_t. In classical classification problems, the value of the correct class in the target vector is set to one, and the rest to zero.

When a new rule is added to the system, its consequent parameters are initialized by the weighted average of the parameters of the other rules, while the parameters of the other rules do not change:

$$\Pi_t = \begin{bmatrix} \boldsymbol{\pi}_{1(t-1)}^1 & \boldsymbol{\pi}_{1(t-1)}^2 & \cdots & \boldsymbol{\pi}_{1(t-1)}^k \\ \boldsymbol{\pi}_{2(t-1)}^1 & \boldsymbol{\pi}_{2(t-1)}^2 & \cdots & \boldsymbol{\pi}_{2(t-1)}^k \\ ... & ... & ... & ... \\ \boldsymbol{\pi}_{r(t-1)}^1 & \boldsymbol{\pi}_{r(t-1)}^2 & \cdots & \boldsymbol{\pi}_{r(t-1)}^k \\ \boldsymbol{\pi}_{(r+1)t}^1 & \boldsymbol{\pi}_{(r+1)t}^2 & \cdots & \boldsymbol{\pi}_{(r+1)t}^k \end{bmatrix} \tag{26}$$

where

$$\boldsymbol{\pi}_{(r+1)t}^c = \sum_{i=1}^{r} \beta_i(\mathbf{x}_t) \boldsymbol{\pi}_{i(t-1)}^c \tag{27}$$

In addition, the covariance matrix C is extended as follows:

$$C_t = \begin{bmatrix} \rho \begin{bmatrix} C_{t-1} \end{bmatrix} & \begin{bmatrix} 0 \end{bmatrix} \\ \begin{bmatrix} 0 \end{bmatrix} & \begin{bmatrix} \Omega & ... & 0 \\ ... & ... & ... \\ 0 & ... & \Omega \end{bmatrix} \end{bmatrix} \tag{28}$$

where $\rho = (r^2 + 1)/r^2$.

The complete learning algorithm can be summarized by Algorithm 1.

Algorithm 1: First-order Takagi-Sugeno online incremental learning algorithm

foreach *new sample* $\mathbf{x}$ **do**
  if $\mathbf{x}$ *is the first sample of a new class* **then**
    create a new fuzzy prototype based at $\mathbf{x}$;
    initialize its potential by 1;
    add a new fuzzy rule to the system;
    extend the consequent parameters matrix as in (26) and (27);
    update and extend the covariance matrix as in (28);
  else
    calculate the activations of the fuzzy rules by (9);
    determine the winning class label by [5];
    get the true class label;
    calculate the potential of $\mathbf{x}$ by (11);
    update the potentials of the existing prototypes centers using (15);
    if $Pot(\mathbf{x}) > Pot(\boldsymbol{\mu}_i) \ \ \forall i \in [1, R]$ **then**
      if $\mathbf{x}$ *is close to an existing center* $\boldsymbol{\mu}_i$ **then**
        let $\mathbf{x}$ be the new center of the prototype P_i and keep the same consequents of the $rule_i$;
      else
        create a new fuzzy prototype based at $\mathbf{x}$;
        initialize its potential by 1;
        add a new fuzzy rule to the system;
        extend the consequent parameters matrix as in (26) and (27);
        update and extend the covariance matrix as in (28);
      end
    end
    Apply premise adaptation according to $\mathbf{x}_b$ by either (16) and (18) or (19) and (20);
    update the consequents parameters using (24) and (25);
  end
end

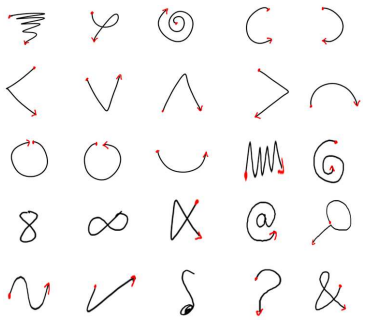


Fig. 2 Handwritten gestures in the dataset SIGN

4 Experimental results

In this section, we analyze the performance and behavior of our model of incremental learning on two types of experience. The first experiment is conducted on an incremental learning problem related to our specific application context, online handwritten gesture recognition. In the second part of this experimental study, we test our model on well-known classification benchmarks, allowing comparing our results with non-incremental learning methods, while offering the opportunity to other incremental approaches to compare with our method.

4.1 Online handwritten gesture recognition

We led the experiments on the "SIGN" database, which is a database of on-line handwritten gestures (figure 2). It is composed of 25 different gestures drawn by 11 different writers on Tablet PCs. Each writer has drawn 100 samples of each gesture, *i.e.* 2,500 gestures in each writer-specific dataset. The dataset (and additional information on the data collection protocol) can be found in [SIGN (2010)]. Each gesture is described by a set of 10 features. The presented results are the average of results of 11 different tests for the 11 writers.

4.2 UCI repository datasets

Besides the SIGN dataset, we evaluate the presented neuro-fuzzy model on some benchmark problems form the UCI machine learning repository [Frank and Asuncion (2010)]. We particularly focus in this work on multi-class problems. We have chosen the next datasets:

- **CoverType**: The aim of this problem is to predict forest cover type from 12 cartographical variables. Seven classes of forest cover types are considered in this dataset. We use in our experiment a subset of 2100 instances.
- **PenDigits**: The objective is to classify the ten digits represented by their handwriting information from pressure sensitive tablet PC. Each digits is represented by 16 features. The dataset contains about 11000 instances.

Consequent Type \ Dataset (#C,#F,#I)		PenDigits (10,16,10992)	CoverType (7,54,2100)	Segment (7,19,2310)	Letters (26,16,20000)	Sign (25,10,2500)
0-Order	Global Learning	4.04 ± 0.41	19.67 ± 1.59	7.90 ± 1.18	12.05± 0.37	2.98 ± 0.51
	Local Learning	4.02 ± 0.47	20.07 ±1.60	7.69 ± 1.05	12.11 ±0.48	3.01 ±0.65
1-Order	Global Learning	3.00 ± 0.36	18.45 ± 1.55	7.00 ± 1.19	10.86 ± 0.49	2.76 ± 0.63
	Local Learning	2.75 ±0.38	18.30 ±1.60	7.17 ± 1.13	11.05 ± 0.49	2.68 ±0.58

Table 1 Misclassification rates for different consequent types

– **Segment**: Each instance in the dataset represents a 3x3 region from 7 outdoor images. The aim is to find the image from which the region was taken. Each region is characterized by 19 numerical attributes. There are 2310 instances in the dataset.
– **Letters**: The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. Each letter is represented by 16 primitive numerical attributes. The dataset contains 20000 instances.

We use 75% of each dataset for the incremental learning process and the rest is used to estimate the performance during and at the end of the learning process. In order to get the results unbiased by the data order effect, we repeat the experiment for each dataset 40 times with different random data orders and the mean results and standard deviations are presented in the figures and the tables. Three incremental learning models are compared in this experiment:

(I) Euclidean + statistical adaptation: an evolving first-order TS classifier with parallel-to-axes hyper-elliptical prototypes (this model is equivalent to eClass model [Angelov et al. (2008)]),
(II) Mahalanobis + statistical adaptation: our extended version with rotated hyper-elliptical prototypes (covariances between features are considered). The adaptation of prototypes is purely statistical (section 3.2.1)
(III) Mahalanobis + confusion-driven adaptation: prototypes adaptation is based on the degrees of confusion (section 3.2.2) in this model.

As for the premise structure, several variants of consequent functions and their learning algorithm can be used as aforementioned. We have measured the impact of these variants on the classification accuracy using the enhanced premise structure on the five datasets. Table 1 shows a comparison between zero-order and first-order model and between local consequent optimization and global consequent optimization, for a hyper-elliptical premise structure with statistical adaptation. We can note that first-order models are more performant than zero-order ones, but with the price of having more parameters. On the other hand, we note that both local and global consequents optimization give almost the same results, with the advantage of having less parameter using the local optimization.

Premise Type \ Dataset (#C,#F,#I)		PenDigits (10,16,10992)	CoverType (7,54,2100)	Segment (7,19,2310)	Letters (26,16,20000)	Sign (25,10,2500)
Euclidean + statistical adaptation (eClass)	0-Order	20.59 ± 0.75	28.77 ± 1.82	21.95±1.71	36.64 ± 0.81	6.05 ±0.88
	1-Order	3.05 ± 0.32	20.46 ±1.38	8.53 ± 1.42	20.22 ±0.57	3.59 ±0.67
Mahalanobis + statistical adaptation	0-Order	4.02 ± 0.47	20.07 ±1.60	7.69 ± 1.05	12.11 ±0.48	3.01 ±0.65
	1-Order	2.75 ±0.38	18.30 ±1.60	7.17 ± 1.13	11.05 ± 0.49	2.68 ±0.58
Mahalanobis + confusion-driven adaptation	0-Order	1.88 ± 0.33	15.71 ±1.70	6.19 ± 0.88	10.05 ±0.35	3.10 ±0.62
	1-Order	1.64 ± 0.31	15.92 ±1.91	6.05 ± 0.95	9.52 ± 0.31	2.08 ±0.57

Table 2 Misclassification rates for different premise structures

Thus, we choose in the next experiment a local consequent optimization algorithm, and we compare the three premise structures that we have mentioned above. Results are shown for both zero-order and first-order models. We note from Table 2 that our improved models outperform the reference model. With the improved structure of premises, the rate of misclassification generally decreases from 10% to 50%, according to the classification problem. We can also notice that the confusion-driven premise adaptation helps in correcting some classification errors and improves the system's performance compared to the statistical adaptation. Almost the same results are obtained for either zero-order or first-order consequents.

We show in figure 3, 4, 5 and 6 the evolution of classification performance during the incremental learning process using the three different premise structures. We show in figure the evolution of classification performance during the incremental learning process using the three different premise structures. We notice the stability and the superiority of the performance using the improved premise structure.

5 Conclusion

An improvement on the premise structure of Takagi-Sugeno neuro-fuzzy classifier had been presented in this paper. The rotated fuzzy zones of influence in the presented model enable the premise "layer" to consider the correlations that may exist between input features and boost the overall performance of the classifier. An appropriate recursive premise adaptation methods are coupled with a known incremental clustering technique and used with the recursive least squares method to learn the classifier in one-pass incremental mode. The proposed enhancements reduce the misclassification rate by more than 30% for our specific classification problem (online handwritten gesture recognition), and these good results are also obtained on several benchmark classification problems.

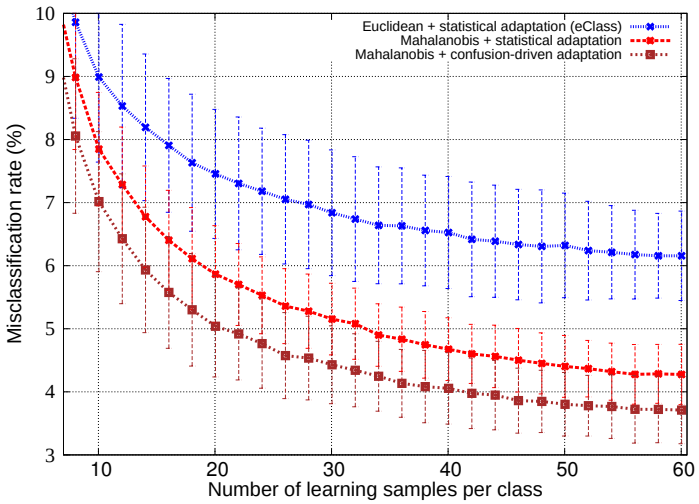


Fig. 3 Evolution of performance during the incremental learning process(SIGN)

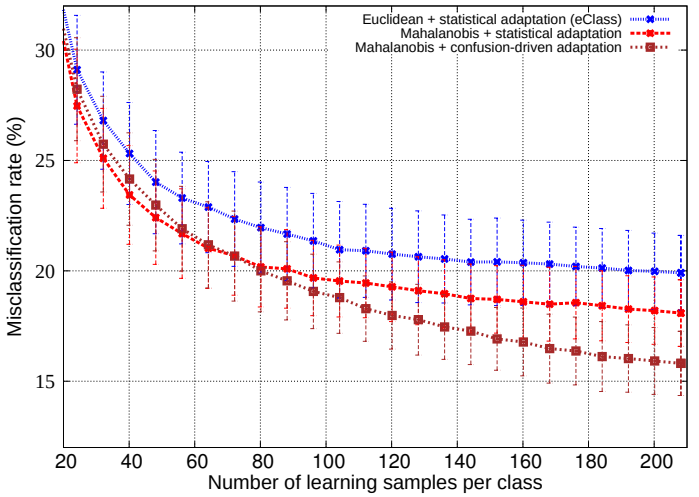


Fig. 4 (CoverType)

References

[SIGN (2010)] http://www.synchromedia.ca/web/ets/gesturedataset.

[Aha et al. (1991)] Aha, D. W., Kibler, D., Albert, M. K., 1991. Instance-based learning algorithms. Mach. Learn. 6 (1), 37–66.

[Angelov (2004)] Angelov, P., 2004. An approach for fuzzy rule-base adaptation using on-line clustering. International Journal of Approximate Reasoning 35 (3), 275 – 289.

[Angelov (2010)] Angelov, P., 2010. Evolving takagi-sugeno fuzzy systems from streaming data, ets+. Evolving Intelligent Systems: Methodology and Applications, 21–50.

[Angelov and Filev (2003)] Angelov, P., Filev, D., 2003. On-line design of takagi-sugeno models. In: IFSA'03: Proceedings of the 10th international fuzzy systems association World Congress conference

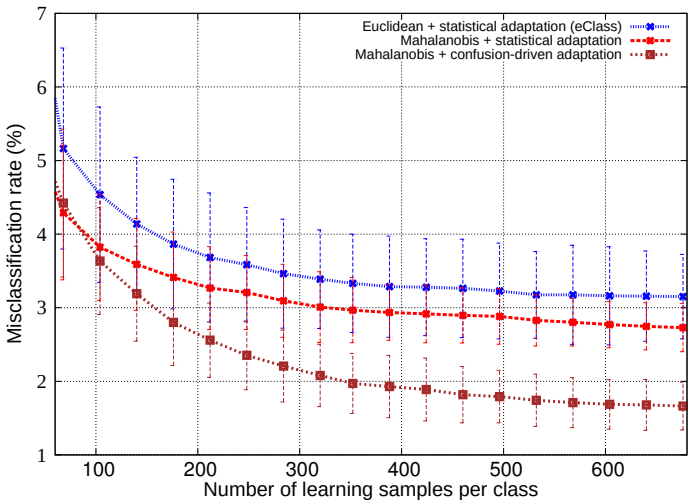


Fig. 5 (PenDigits)

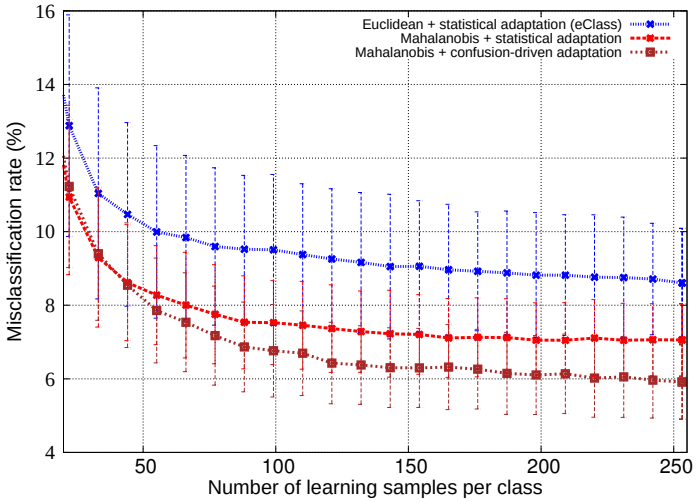


Fig. 6 (Segment)

on Fuzzy sets and systems. pp. 576–584.

[Angelov and Filev (2004)] Angelov, P., Filev, D., Feb. 2004. An approach to online identification of takagi-sugeno fuzzy models. IEEE Transactions on Systems, Man, and Cybernetics 34 (1), 484–498.

[Angelov et al. (2008)] Angelov, P., Lughofer, E., Zhou, X., 2008. Evolving fuzzy classifiers using different model architectures. Fuzzy Sets Syst. 159 (23), 3160–3182.

[Angelov and Zhou (2006)] Angelov, P., Zhou, X., 2006. Evolving fuzzy systems from data streams in real-time. In: IEEE Symposium on Evolving Fuzzy Systems.

[Carpenter et al. (1992)] Carpenter, G., Grossberg, S., Markuzon, N., Reynolds, J., Rosen, D., 1992. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multi-dimensional maps. IEEE Transactions on Neural Networks 3.

[Carpenter and Grossberg (1988)] Carpenter, G. A., Grossberg, S., 1988. The art of adaptive pattern recognition by a self-organizing neural network. Computer 21 (3), 77–88.

[De Backer and Scheunders (2001)] De Backer, S., Scheunders, P., 2001. Texture segmentation by frequency-sensitive elliptical competitive learning. Image and Vision Computing 19 (9–10), 639–648.

[Frank and Asuncion (2010)] Frank, A., Asuncion, A., 2010. UCI machine learning repository. URL `http://archive.ics.uci.edu/ml`

[Gary G. (2001)] Gary G. Yen, 2001. An effective neuro-fuzzy paradigm for machinery condition health monitoring. IEEE Transactions on Systems, Man, and Cybernetics 31-4, 523 – 536.

[Jang (1993)] Jang, J.-S., 1993. Anfis: adaptive-network-based fuzzy inference system. Systems, Man and Cybernetics, IEEE Transactions on 23, 665–685.

[Kasabov (2007)] Kasabov, N., 2007. Evolving connectionist systems: The knowledge engineering approach (2nd Ed.). Springer.

[Littlestone (1991)] Littlestone, N., 1991. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. In: COLT '91: Proceedings of the fourth annual workshop on Computational learning theory. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 147–156.

[Lughofer (2008)] Lughofer, E., 2008. Flexfis: A robust incremental learning approach for evolving takagi-sugeno fuzzy models. IEEE T. Fuzzy Systems 16 (6), 1393–1410.

[Mouchere et al. (2007)] Mouchere, H., Anquetil, E., Ragot, N., 2007. On-line writer adaptation for handwriting recognition using fuzzy inference systems. International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) 21(1), 99–116.

[Polikar et al. (2001)] Polikar, R., Udpa, L., Udpa, S., Honavar, V., 2001. Learn++: An incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, Man, and Cybernetics 31, 497–508.

[Reinke and Michalski (1988)] Reinke, R., Michalski, R., 1988. Incremental learning of concept descriptions: A method and experimental results. Hayes, J., Michie, D., Richards, J., eds.: Machine Intelligence 11, 263288.

[Sadri et al. (2006)] Sadri, J., Suen, C. Y., Bui, T. D., 2006. A new clustering method for improving plasticity and stability in handwritten character recognition systems. Pattern Recognition, International Conference on 2, 1130–1133.

[Song and Kasabov (2000)] Song, Q., Kasabov, N., 2000. Dynamic evolving neuro-fuzzy inference system (denfis): On-line learning and application for time-series prediction. IEEE Transactions of Fuzzy Systems 10, 144–154.

[Yager and Filev (1993)] Yager, R. R., Filev, D. P., 1993. Learning of fuzzy rules by mountain clustering. Vol. 2061. SPIE, pp. 246–254.

[Zwickel and Wills (2005)] Zwickel, J., Wills, A. J., 2005. New Directions in Human Associative Learning. Psychology Press, Ch. Integrating associative models of supervised and unsupervised categorization, p. 118.