HAL open science

Decremental Learning of Evolving Fuzzy Inference Systems, Application to Handwritten Gestures Recognition

Manuel Bouillon, Eric Anquetil, Abdullah Almaksour

▶ **To cite this version:**

Manuel Bouillon, Eric Anquetil, Abdullah Almaksour. Decremental Learning of Evolving Fuzzy Inference Systems, Application to Handwritten Gestures Recognition. 9th International Conference on Machine Learning and Data Mining (MLDM), Jul 2013, New-York, United States. pp.115-129, 10.1007/978-3-642-39712-7_9 . hal-00821990

HAL Id: hal-00821990

https://inria.hal.science/hal-00821990v1

Submitted on 13 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Decremental Learning of Evolving Fuzzy Inference Systems

Application to Handwritten Gestures Recognition

Manuel Bouillon*, Eric Anquetil, and Abdullah Almaksour

Université Européenne de Bretagne, France
INSA de Rennes, Avenue des Buttes de Coesmes, F-35043 Rennes
IRISA, Campus de Beaulieu, F-35042 Rennes
{Manuel.Bouillon,Eric.Anquetil,Abdullah.Almaksour}@irisa.fr
http://www.irisa.fr/intuidoc/

Abstract. This paper tackles the problem of incremental and decremental learning of an evolving and customizable fuzzy inference system for classification. We explain the interest of integrating a forgetting capacity in such an evolving system to improve its performances in changing environment. In this paper, we describe two decremental learning strategies, to introduce a forgetting capacity in evolving fuzzy inference systems. Both techniques use a sliding window to introduce forgetting in the optimization process of fuzzy rules conclusions. The first approach is based on a downdating technique of least squares solutions for unlearning old data. The second integrates differed directional forgetting in the covariance matrices used in the recursive least square algorithm. These techniques are first evaluated on handwritten gesture recognition tasks in changing environments. They are also evaluated on some well-known classification benchmarks. In particular, it is shown that decremental learning allow to adapt to concept drifts. It is also demonstrated that decremental learning is necessary to maintain the system capacity of learning new classes over time, making decremental learning essential for the life-time use of an evolving and customizable classification system.

Key words: Online Classification; Incremental Learning; Decremental Learning; Evolving Fuzzy Inference System; Recursive Least Squares; Concept Drifts; Forgetting

1 Introduction

Evolving classification systems have appeared in the last decade to meet the need for recognizers that work in changing environments. They use incremental learning to adapt to the data flow and to cope with class adding (or removal) at run time. This paper focuses on integrating a forgetting capacity in evolving fuzzy inference systems to improve their performances in changing environments.

* Corresponding author

The aim of that forgetting capacity is twofold: first, maintain the system capacity of learning new classes over time, and second, enable the system to follow changes of its environment (so-called concept drifts).

The target application of this work is the use of online handwritten gesture classifiers to facilitate user interactions[1] on pen-based interfaces like tablet computers, smart-phones, whiteboards, etc. Gestures can be drawn differently from one user to another, and users may want to add or remove gestures, as long as they use the application. Moreover, users would often change progressively the manner by which they draw gestures. Novice users start drawing carefully and slowly their gestures, while they do them in a more fluid and rapid manner as they become expert. The classifier hence needs to evolve and follow the changes in the data flow. If most users will use a common subset of gestures, each user will need some specific gestures classes for his own usage but that others won't use. In addition, classifier usage may change with time, and the end user may need to add, remove or change gestures classes to fit his needs. That is why the classifier needs to be customizable by end users. To cope with these requirements, a forgetting capacity must be used to increase system reactivity and performances in such dynamic environments.

In this paper, we extend our evolving classification system *Evolve* [1] by integrating a forgetting capacity with the use of decremental learning. Two new decremental learning strategies are presented, both relying on a sliding window of data samples. The first technique uses this window to completely unlearn old data, by downdating the least squares solutions. The second technique uses this window to integrate differed directional forgetting in the learning process, and allow old data to be forgotten if needed.

We briefly present the architecture of *Evolve* and its incremental learning algorithm in Section 2. Section 3 present forgetting interest and existing techniques. Our two new approaches are presented in Section 4 and 5. These approaches are then evaluated on some handwritten gesture recognition tasks in section 6. Section 7 concludes and discusses future work.

2 System architecture

We focus here on Fuzzy Inference Systems (FIS) [12], with first order conclusion structure [15]. FIS have demonstrated their good performances for incremental classification of changing data flows [2]. Moreover, they can easily be trained online (in real time) and have a good behavior when new classes are added. [2] and [1] are recent examples of evolving FIS used for online classification.

Fuzzy inference systems consist of a set of fuzzy inference rules like the following rule example.

$$\mathbf{Rule}^{(i)}: \ \mathbf{IF}\ \mathbf{x} \text{ is close to } C^{(i)}\ \mathbf{THEN}\ \hat{\mathbf{y}}^{(i)} = (\hat{\mathbf{y}}_1^{(i)}; \ldots; \hat{\mathbf{y}}_c^{(i)})^\top \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the feature vector, $C^{(i)}$ the fuzzy prototype associated to the i-th rule and $\hat{\mathbf{y}}^{(i)\top} \in \mathbb{R}^c$ the output vector. Rule premises are the fuzzy membership

[1] See `http://youtu.be/qOx4IY6uYf8` for a demonstration of gestural commands

to rule prototypes, which are clusters in the input space. Rule conclusions are fuzzy membership to all classes, that are combined to produce the system output.

2.1 Premise Structure

Our model uses rotated hyper-elliptical prototypes that are each defined by a center $\boldsymbol{\mu}^{(i)} \in \mathbb{R}^n$:

$$\boldsymbol{\mu}^{(i)} = (\mu_1^{(i)}; \dots, \mu_n^{(i)})^\top \tag{2}$$

and covariance matrix $\Sigma^{(i)} \in \mathbb{R}^{n \times n}$ (where n is the number of features):

$$\Sigma^{(i)} = \begin{bmatrix} \sigma_1^2 & \dots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{n,1} & \dots & \sigma_n^2 \end{bmatrix} \tag{3}$$

To measure the activation degree of each fuzzy prototype, we use the multivariate normal distribution:

$$\alpha^{(i)}(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma^{(i)}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top (\Sigma^{(i)})^{-1} (\mathbf{x} - \boldsymbol{\mu}^{(i)})^\top\right) \tag{4}$$

2.2 Inference Process

The inference process consist of three steps:

1. Activation degree is computed for every rule with Equation 4, and then normalized as follow:
$$\alpha^{(i)}(\mathbf{x}) = \frac{\alpha^{(i)}(\mathbf{x})}{\sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x})} \tag{5}$$
where r is the number of rules.
2. System output is obtained from rule outputs using sum-product inference:
$$\hat{\mathbf{y}} = \sum_{k=1}^{r} \alpha^{(k)}(\mathbf{x}) \cdot \hat{\mathbf{y}}^{(k)} \tag{6}$$
3. Predicted class is the one corresponding to the highest output:
$$\text{class}(\mathbf{x}) = \arg\max_{k=1}^{c}(\hat{\mathbf{y}}_k) \tag{7}$$

2.3 Conclusion Structure

In a first order FIS, rule conclusions are linear functions of the inputs:

$$\hat{\mathbf{y}}^{(i)\top} = (l_1^{(i)}(\mathbf{x})\,; \dots ;\, l_c^{(i)}(\mathbf{x})) \tag{8}$$

$$l_k^{(i)}(\mathbf{x}) = \mathbf{x}^\top \cdot \boldsymbol{\theta}_k^{(i)} = \theta_{0,k}^{(i)} + \theta_{1,k}^{(i)} \cdot x_1 + \dots + \theta_{n,k}^{(i)} \cdot x_n \tag{9}$$

The i-th rule conclusion can be reformulated as:

$$\hat{\mathbf{y}}^{(i)\top} = \mathbf{x}^\top \cdot \Theta^{(i)} \tag{10}$$

with $\Theta^{(i)} \in \mathbb{R}^{n \times c}$ the matrix of the c linear function coefficients of the i-th rule:

$$\Theta^{(i)} = (\boldsymbol{\theta}_k^{(i)}; \ldots; \boldsymbol{\theta}_c^{(i)}) = \begin{bmatrix} \theta_{1,1}^{(i)} & \ldots & \theta_{1,c}^{(i)} \\ \vdots & \ddots & \vdots \\ \theta_{n,1}^{(i)} & \ldots & \theta_{n,c}^{(i)} \end{bmatrix} \tag{11}$$

2.4 Incremental Learning Process

Let $\mathbf{x}_i$ $(i = 1..t)$ be the i-th data sample, M_i the model at time i, and f the learning algorithm. The incremental learning process can be defined as follow:

$$M_i = f(M_{i-1}, \mathbf{x}_i) \tag{12}$$

whereas a batch learning process would be:

$$M_i = f(\mathbf{x}_1, \ldots, \mathbf{x}_i) \tag{13}$$

In our recognizer *Evolve* [1], both rule premises and conclusions are incrementally adapted:

1. Rule prototypes are statistically updated to model the runtime data:

$$\boldsymbol{\mu}_t^{(i)} = \frac{(t-1) \cdot \boldsymbol{\mu}_{t-1}^{(i)} + \mathbf{x}_t}{t} \tag{14}$$

$$\Sigma_t^{(i)} = \frac{(t-1) \cdot \Sigma_{t-1}^{(i)} + (\mathbf{x}_t - \boldsymbol{\mu}_{t-1}^{(i)})(\mathbf{x}_t - \boldsymbol{\mu}_t^{(i)})}{t} \tag{15}$$

2. Rule conclusions parameters are optimized on the data flow, using Recursive Least Squares (RLS) algorithm [9]:

$$\Theta_t^{(i)} = \Theta_{t-1}^{(i)} + \alpha^{(i)} \mathrm{C}_t^{(i)} \mathbf{x}_t (\mathbf{y}_t^\top - \mathbf{x}_t^T \Theta_{t-1}^{(i)}) \tag{16}$$

$$\mathrm{C}_t^{(i)} = \mathrm{C}_{t-1}^{(i)} - \frac{\mathrm{C}_{t-1}^{(i)} \mathbf{x}_t \mathbf{x}_t^\top \mathrm{C}_{t-1}^{(i)}}{\frac{1}{\alpha^{(i)}} + \mathbf{x}_t^\top \mathrm{C}_{t-1}^{(i)} \mathbf{x}_t} \tag{17}$$

New rules, with their associated prototypes and conclusions, are created by the incremental clustering method *eClustering* [3] when needed.

3 Decremental Learning

This paper focuses on the decremental learning of an evolving fuzzy inference system optimized with the Recursive Least Squares (RLS) algorithm.

Introducing forgetting in Recursive Least Square (RLS) is a well studied problem. The principle of forgetting in the RLS algorithm is to prevent the covariance matrix, which can be seen as a representation of the system gain, to go to zero (but without making it going to infinity either).

Interest of Forgetting. The interest of integrating forgetting in a learning system is twofold. First, a forgetting capacity is necessary to limit the weight of past data, and thus to maintain system capacity of learning new classes. If every data sample has the same relative weight, learning gain will tend to zero and system will tend to become set with time. Second, a forgetting capacity allows the system to follow any change – concept drift [17] – of the data flow by forgetting obsolete data.

On the other side, we mustn't forget too much. Our purpose here is to improve the behavior of our system in non-stationary scenarios, but we mustn't reduce our system performances in stationary scenarios, nor make our system collapse by forgetting too much – so-called "catastrophic forgetting".

Existing techniques. Several forgetting techniques for the Recursive Least Square (RLS) algorithm already exist, like in the literature dedicated to control of complex systems. The most common approach is to introduce an exponential weighting of data with time by the use of an exponential forgetting factor [10]. It comes to using the RLS algorithm on an (exponentially weighted) sliding window which seems a good idea.

However, this algorithm behaves poorly when systems are not uniformly exited – it is known as the covariance "wind-up" problem [11] – which is the case in classification problems.

This problem happens when some of the covariance matrix elements grows abnormally and make the estimator overstep and diverge from its optimum value. This "wind up" problem is due to the fact that this exponential factor produce an uniform forgetting whereas the excitation of the system varies with time, and is not uniform over the input space. Several "had-oc" strategies [5], [14], [8] have been proposed to deal with this instability problem but no universal solution.

Proposed approaches The idea of using a sliding window is not new, it has been extensively used as a way to get a reactive estimator that follows concept drifts. We use this window to obtain two new decremental learning strategies applied to FIS optimization with the RLS algorithm. The first approach is based on downdating the least square solutions, which unlearn "old" data that leave the sliding window. The second approach is based on differed directional forgetting of the covariance matrix used in the RLS algorithm. The length of this window is a sensitive issue that will be discussed in Section 5.

4 A First Approach: Downdating Least Square Solutions

The principle of this approach is simple, we maintain a sliding window over the latest data, and we optimize the rules conclusions only on this window of data.

As we can't afford to rebuild rules conclusions at the arrival of each new data, they are updated using the recursive least squares (RLS) algorithm [1]. In the same way, we here downdate least squares solutions at the departure of each old data from the window, without complete rebuilding. To do so, we use the de-recursive least squares algorithm in order to recursively unlearn old data.

De-Recursive Least Squares (DRLS) Algorithm. Let $\Theta^{(i)}_{s\to t} \in \mathbb{R}^{n\times c}$ be the i-th rule conclusion, optimized on data samples from $(\mathbf{x}_s, \mathbf{y}_s)$ to $(\mathbf{x}_t, \mathbf{y}_t)$ (with $\mathbf{x}_k \in \mathbb{R}^n$ the input vectors, and $\mathbf{y}_k \in \mathbb{R}^c$ the binary output vectors).

We propose to downdate least squares solutions with the following DRLS formulas to unlearn "old" data sample $(\mathbf{x}_s, \mathbf{y}_s)$.

$$\Theta^{(i)}_{(s+1)\to t} = \Theta^{(i)}_{s\to t} - \alpha^{(i)}_s \mathrm{C}^{(i)}_{(s+1)\to t}\mathbf{x}_s(\mathbf{y}_s^\top - \mathbf{x}_s^\top \Theta^{(i)}_{s\to t}) \tag{18}$$

Where the covariance matrices $\mathrm{C}^{(i)}$ are updated as follows.

$$\mathrm{C}^{(i)}_{(s+1)\to t} = \mathrm{C}^{(i)}_{s\to t} + \frac{\mathrm{C}^{(i)}_{s\to t}\mathbf{x}_s\mathbf{x}_s^\top \mathrm{C}^{(i)}_{s\to t}}{\frac{-1}{\alpha^{(i)}_s} + \mathbf{x}_s^\top \mathrm{C}^{(i)}_{s\to t}\mathbf{x}_s} \tag{19}$$

These DRLS formulas are proven in Appendix A.

This real time techniques gives the same results as a batch learning on the window of data. As a result, the learning gain is kept bounded and the system keeps its learning properties over time. New classes are learned as quickly after a long learning time as after a short one. Moreover, this sliding window enable the system to easily adapt to concept drifts. Data samples that leave the window are unlearned, and system quickly adapt to new concept as old one is unlearned.

5 A Second Approach: Differed Directional Forgetting

We present here a simple but very efficient new type of forgetting, without any exponential factor, and based on past data. Our strategy is to use forgetting but in the direction of "old" data samples. As for the previous approach, we use a sliding window that enable us to remove "old" data weight, when they leave the window. The weight removal can be done using the second de-recursive formula for the covariance matrix downdating (Equation 19).

$$\mathrm{C}^{(i)}_{(s+1)\to t} = \mathrm{C}^{(i)}_{s\to t} + \frac{\mathrm{C}^{(i)}_{s\to t}\mathbf{x}_s\mathbf{x}_s^\top \mathrm{C}^{(i)}_{s\to t}}{\frac{-1}{\alpha^{(i)}_s} + \mathbf{x}_s^\top \mathrm{C}^{(i)}_{s\to t}\mathbf{x}_s}$$

From Unlearning to Forgetting. The de-recursive least square algorithm enable us to unlearn, to remove completely the effect some data have had in the optimization process. This unlearning take place in two steps: a first step to downdate the covariance matrix and a second step to downdate the rule conclusion coefficients.

Downdating the covariance matrix is indispensable in order to limit old data weight. Unlearning rule conclusions make the system change and follow concept drifts. However, if downdating the covariance matrix has little impact on the system recognition performances, unlearning rule conclusions deconstructs the conclusion matrix – erases the knowledge previously acquired – and reduces the system performances.

The learning of the conclusions at the arrival of a data sample is a step forward, their unlearning is a step backward. At the arrival of a new sample, this step backward will be redone, but with some minor adjustments. It is more interesting not to step backward but to wait and sidestep instead when a new data sample arrives.

Following this philosophy, the second approach only update the covariance matrix, and not the rule conclusions. By doing so, we go from a downdating approach to a forgetting one. Old knowledge isn't erased, but will be overwritten by future knowledge since no more weight is given to the corresponding data in the covariance matrix.

Discussion of the Window Length

The sensitive issue of these two approaches is the length of the sliding window. Indeed, performances are directly linked to the window length. In steady environment, the longer is the window, the lower is the error rate (until a minimum rate). However, a too long window reduces system reactiveness and thus deteriorates performances in changing environment.

We focus on having a large enough window to avoid performances reduction in stationary environments. Such window is sufficient to limit old data weight and to adapt to concept drifts in changing environments. Even if a fixed length sliding window is not optimal, it provides a good behavior as it will be demonstrated experimentally.

The window minimum length is a problem dependent variable that can be empirically determined. It depend on the number of classes, the set of features used and the intrinsic difficulty of the classification problem. Nevertheless, a satisfying window length can easily be found experimentally as will be shown in the next section.

A drawback of these approaches is the need for memorizing all the data in the window to be able to unlearn/forget them. However, the increase in memory requirements remains quite small, compared to the initial algorithm, as will be shown in the next section.

6 Experimental Results

Starting from the state-of-the-art recognizer *Evolve* [1], we implemented our two new approaches: *Evolve D* – with Downdating – and *Evolve F* – with Forgetting.

As this work is applied to online handwritten gesture recognition, we first evaluate our two new systems, and the reference one, on handwritten gestures recognition tasks in changing environments. Then, we evaluate these two approaches on common classification benchmark datasets.

Incremental Evaluation Protocol. To evaluate our systems in a realistic way, we used an incremental evaluation protocol called predictive sequential – or prequential [7] – with a sliding window to converge to the holdout error.

As an incremental system first tries to recognize a data sample, and then learn from it once it has the true label, we evaluated our systems in a similar way. Each data sample is first used as test sample, and then as learning sample. Error rates are then computed between every test points.

6.1 Evaluation on Gestures Recognition Tasks

As this work is applied to online handwritten gesture recognition, we evaluated our new approaches on two handwritten gestures databases: ILGDB[2] [13] and IRONOFF-digits [16].

The Heterogeneous Baseline Feature set (HBF49) [4] and a sliding window length of 50 data samples is used in both cases. Such a window length (with HBF49) only require to memorize a 50 by 50 matrix, which correspond to the size of the covariance matrix of one rule.

ILGDB This database contains handwritten gestures that have been collected in an immersive environment. It is composed of 6629 mono-stroke gestures, belonging to 21 classes, which were written by 38 writers. This database is very interesting for several reasons.

First, gestures are ordered chronologically in their drawing order which allows us to see changes in writer style with time, as the writer changes from novice to expert. Second, class frequencies varies, from 5 to 17 examples per class.

Third, for part of the database, gesture classes are user defined. This features makes this database very realistic and representative of the real use of an online recognition system. Some gesture examples are shown in figure 1.

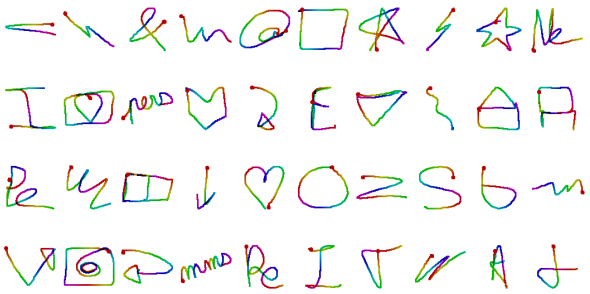

Fig. 1. Gesture samples from ILGDB group 1 (free gestures)

Ironoff-digits The interest of this database is that it offers more gestures (in writer independent mode), but like most classic benchmark databases, IRONOFF-digits is not ordered (no chronological evolution of the data with time).

[2] Freely available at http://www.irisa.fr/intuidoc/ILGDB.html

System Inertia to Novelty (New Classes). We test the reactiveness of our systems, and the evolution of the inertia of their model with time. To that purpose, we train the different systems with seven classes during a varying period time (70, 700 and 2100 training samples) and then introduce three other classes. We measure the time needed by the different systems to learn those new classes. Results averaged over 20 different data orders are shown Figure 2.

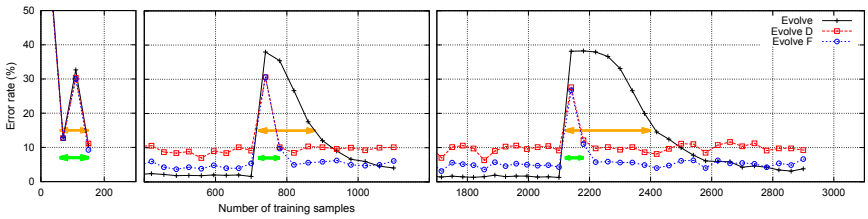

Fig. 2. Scenario "inertia and reactiveness" – IRONOFF-digits database – Adding new classes after different training times

If all systems behave similarly after 70 training samples, results are different after 700 or 2100 samples.

Looking at the results of our reference system without forgetting *Evolve*, we can clearly see that its reactiveness decreases with time, and that the model tends to become set. After 2100 training samples, the system needs 400 more samples to resume to an error rate under 10%, which makes it partially unusable for quite some time.

On the contrary, both systems using decremental learning need the same time to learn and adapt their model to the novelty, whenever it is introduced. Their reactiveness is independent of their age.

This test scenario shows clearly the necessity of integrating a forgetting capacity into an incremental learning system operating in a changing environment.

Performances in Slow Changing Environment. We test our systems on a scenario simulating slow concept drifts. For this purpose, we used the ILGDB 11 writers of group 3 (whose gestures for each class are identical) in a row. We computed the error rate for each writer, without taking into account the first 3 samples per class per writer, to measure system performance when concept drifts are learned. Mean results over 100 writer orders are plotted Figure 3.

Writers, and their drawing style, changes but the base gestures of each class stay unchanged. This test scenario is thus nearly stationary and doesn't really require the use of decremental learning. Our reference system *Evolve* achieves quite good results here with an average error rate of 6.31%. Without forgetting, the system is learning from every writers and become a writer independent recognizer.

Evolve D obtains an average error rates of 11.25%. Its performances are limited by the restrained number of data it is learning from (as old data samples

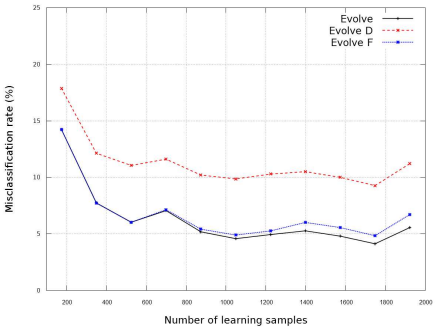

Fig. 3. Performances in slow changing environment – Using ILGDB 11 writers of group 3 (fixed gestures) in a row

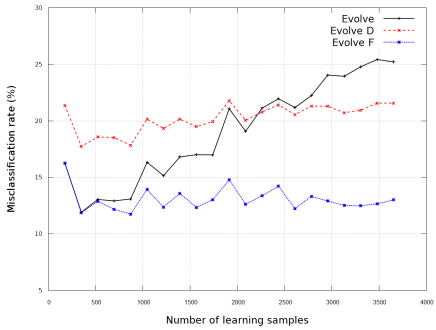

Fig. 4. Performances in fast changing environment – Using ILGDB 21 writers of group 1 (free gestures) in a row

are completely unlearned). The use of downdating prevent the system to build a writer independent model and force it to adapt to the current writer (with very few data).

Evolve F reaches an average error rates of 6.70% which is nearly as good as *Evolve*. This is due to the fact that old data samples are not unlearned, but only their corresponding weights are removed from the covariance matrix used in the RLS algorithm. This forgetting allow the system to keep previously acquired knowledge until it is overwritten by some new information.

Performances in Fast Changing Environment. We also test our systems on a scenario simulating fast concept drifts. For this purpose, we used the ILGDB 21 writers of group 1 (whose gestures are different for the same class) in a row. We computed the error rate for each writer, without taking into account the first 3 samples per class of each writer, to measure system performance after concept drifts are learned. Mean results over 100 writer orders are plotted Figure 4.

This testing scenario is quite difficult because the gestures of every classes completely change as the writer changes. The use of decremental learning is here mandatory.

The results on this scenario are sharply contrasted. Without any forgetting capacity, *Evolve* is not able to adapt its model to the changes in the data flow and ends up with an error rate of 25%.

Evolve D is again limited by the few data its model is build from. Although stable, its performances are quite poor: 20.26 % error rate in average.

Evolve F, maintains reasonable performances with an average error rate of 13.08 %. The use of (differed directional) forgetting allow *Evolve F* to follow the fast concept drifts.

6.2 Evaluation on UCI Benchmark Datasets

We also evaluated our new methods on well-known classification benchmarks from the UCI machine learning repository[6] to test our methods outside the field of handwritten gesture classification and with other feature sets. We chose some datasets following two criteria. First, they are multi-classes problems (our systems are optimized for classification problem with more than two classes), and second, they are large enough datasets and allow the incremental learning and testing of our systems on the long run. We chose the Pen-Digits and Japanese-Vowels datasets.

In this context, the size of the sliding window has to be changed to adapt to those problems difficulty. We used a window length of 500 samples which empirically gives the best results. It increases the memory requirements of our systems (by storing a 500 by 14 or 16 matrix) but is necessary to keep good performances for those more difficult problems.

Japanese-Vowels The Japanese-Vowels dataset is composed of about 10'000 samples. This dataset contains 9 classes: 9 different male speakers that must be recognized using 14 features extracted from two Japanese vowels.

Pen-Digits The Pen-Digits dataset contains about 11'000 handwritten digits that were recorded on a pen based interface. This dataset contains 10 classes and uses 16 features.

Performances in Stationary Environment. We evaluated our two new approaches in stationary environment with the Pen-Digits and Japanese-Vowels datasets to see the effects of decremental learning when it is not needed. Results averaged on 100 different data orders are presented Figure 5

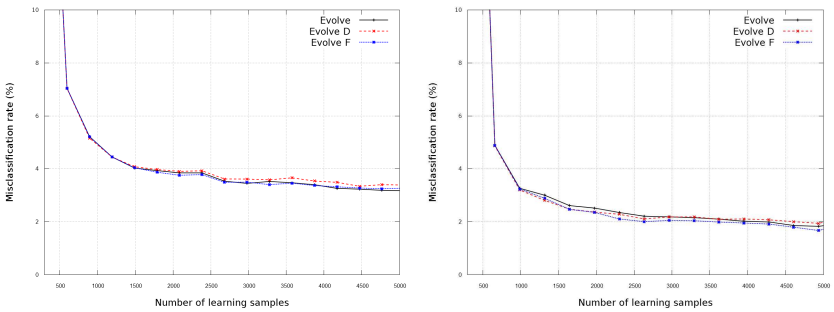

Fig. 5. Performances on Japanese-Vowels and Pen-Digits datasets (UCI repository) in stationary environment

If the use of decremental learning doesn't improve performances, which is quite normal in stationary environment, it doesn't deteriorate them either.

Performances in Changing Environment. As the aim of decremental learning is to improve performances in changing environment, we simulated one by adding half of the classes in the middle of the test. Results averaged on 100 different data orders are presented Figure 6

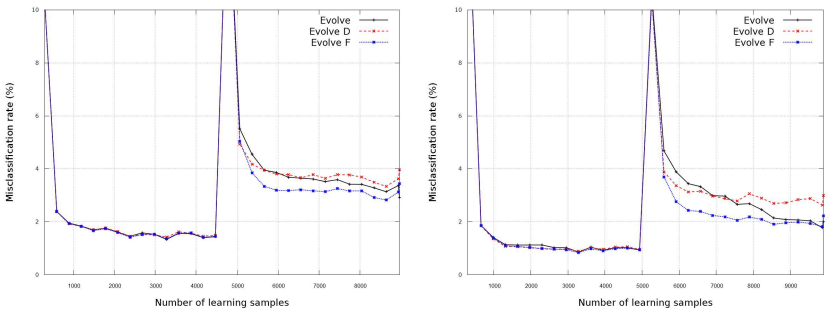

Fig. 6. Performances on Japanese-Vowels and Pen-Digits datasets (UCI repository) in changing environment (half of the classes are added in the middle of the test)

In this changing environments, decremental learning allow to adapt faster to the adding of new classes. Using downdating (*Evolve D*) provide a limited gain as final performances are limited but using forgetting (*Evolve F*) allow to reach final performance faster. *Evolve F* reaches final performances in 1500 and 3000 samples respectively whereas *Evolve* needs 4500 and 5000 samples respectively to converge (on Japanese Vowels and Pen Digits respectively).

6.3 Results Discussion

The performance of the first approach – downdating (*Evolve D*) – are quite limited compared to the second approach – forgetting (*Evolve F*). When downdating, we unlearn completely "old" data and deteriorate least squares solutions. On the other hand, when forgetting, the least squares solutions aren't modified. "Old" data weight is removed, but the knowledge stemming from them is kept until any new data make it mandatory to overwrite it.

On stationary scenarios, decremental learning isn't necessary. *Evolve F*, using (differed directional) forgetting, obtain similar recognition rates than *Evolve*, our reference model. The performance of *Evolve D*, using downdating, are not as good due to the limited number of samples its model is build from.

Results on non-stationary scenarios are quite clear-cut. They show the necessity of using decremental learning to maintain the reactiveness of the system over time. Without downdating, the system model tends to become set, and take ages to learn some novelty. Decremental learning is necessary to maintain system ability to learn new classes over time.

In the same way, we showed that decremental learning is essential to face concept drifts. Without downdating, the system model becomes more and more

complex and performance slowly but surely collapses. Decremental learning allows to discard obsolete data and thus enables the system to focus on current system environment.

These features are very important as our goal is to obtain an evolving classifier to facilitate user interaction on touch sensitive interfaces. Use-cases of touch sensitive interfaces are various and aplenty, so it is essential to give the user the ability to customize the classifier to his needs and adapt to its changes of use.

7 Conclusion

In this paper, we investigated two decremental learning strategies, to introduce a forgetting capacity in evolving fuzzy inference system used for classification.

Both techniques use a sliding window to introduce forgetting in the optimization process of fuzzy rules conclusions. The first approach is based on a downdating technique of least squares solutions for unlearning old data. The second integrates differed directional forgetting in the covariance matrices used in the recursive least square algorithm.

This work is applied on handwritten gesture recognition as our goal is to obtain an evolving and customizable classifier to facilitate user interaction on touch sensitive interfaces. Such an applicative context in clearly non-stationary and require the classifier to allow class adding and to follow concept drift to adapt to its use.

We showed that our second approach *Evolve F* – differed directional forgetting – performs well in changing environment, without deteriorating performance in stationary environment. Our method produce a similar effect than existing techniques to introduce forgetting in the recursive least square algorithm, but without the problem of estimator "wind up".

Future Work A judicious improvement to this method would be to manage the window size adaptively. The window length could be increased as long as it improve performances, and reduced when concept drifts are detected.

Another direction that should be explored is managing adaptively the content of the sliding window. It could be interesting to more data samples of the classes that are harder to recognize, or more subject to confusion with other classes, and less samples of the easier classes.

A Appendix: Proof of De-Recursive Least Squares

Let $\Theta^{(i)}_{s\to t} \in \mathbb{R}^{n\times c}$ be the ith rule conclusion, optimized on data samples from $(\mathbf{x}_s, \mathbf{y}_s)$ to $(\mathbf{x}_t, \mathbf{y}_t)$ (with $\mathbf{x}_k \in \mathbb{R}^n$ the input vectors, and $\mathbf{y}_k^\top \in \mathbb{R}^c$ the binary output vectors).

A.1 Downdating Least Square Solutions

The rule cost functions are (for $1 \leq i \leq r$, where r is the number of rules)

$$J^{(i)}_{s\to t} = \sum_{i=s}^{t} \alpha^{(i)} (\mathbf{y}_i - \mathbf{x}_i^T \Theta^{(i)}_t)^2 \tag{20}$$

$$J^{(i)}_{s\to t} = (\mathrm{Y}_{s\to t} - \mathrm{X}_{s\to t}\Theta^{(i)}_{s\to t})^\top \mathrm{A}^{(i)}_{s\to t}(\mathrm{Y}_{s\to t} - \mathrm{X}_{s\to t}\Theta^{(i)}) \tag{21}$$

where $\mathrm{X}_{s\to t} = [\mathbf{x}_s; \ldots; \mathbf{x}_t]^\top$ with $\mathbf{x}_k = [x_{k,1}; \ldots; x_{k,n}]^\top$ and n the number of dimensions of the input space ; $\mathrm{Y}_{s\to t} = [\mathbf{y}_s^\top; \ldots; \mathbf{y}_t^\top]^T$ with $\mathbf{y}_k = [y_{k,1}; \ldots; y_{k,n}]$ and $y_{k,l} = 1$ if $y_{k,l}$ belong to class l and $y_{k,l} = 0$ otherwise. The weighting matrices $\mathrm{A}^{(i)}_{s\to t}$ are defined as

$$\mathrm{A}^{(i)}_{s\to t} = \alpha^{(i)}_s \cdot \mathrm{Id} \tag{22}$$

with $\alpha^{(i)}_k$ the firing level of the ith rule by data sample x_k, and Id the identity matrix. The weighted least squares solutions are

$$\Theta^{(i)}_{s\to t} = \mathrm{C}^{(i)}_{s\to t}\mathrm{X}^\top_{s\to t}\mathrm{A}^{(i)}_{s\to t}\mathrm{Y}_{s\to t} \tag{23}$$

with the information matrices

$$R^{(i)}_{s\to t} = (\mathrm{C}^{(i)}_{s\to t})^{-1} = \mathrm{X}_{s\to t}\mathrm{A}^{(i)}_{s\to t}\mathrm{X}^\top_{s\to t} \tag{24}$$

The information matrices can easily be downdated

$$\mathrm{R}^{(i)}_{(s+1)\to t} = \mathrm{R}^{(i)}_{s\to t} - \mathbf{x}_s \alpha^{(i)}_s \mathbf{x}_s^\top \tag{25}$$

and using the matrix inversion lemma (Woodbury identity)

$$\left(A + XBX^\top\right)^{-1} \tag{26}$$
$$= A^{-1} - A^{-1}X\left(B^{-1} + X^\top A^{-1}X\right)^{-1} X^\top A^{-1}$$

with $A = \mathrm{R}^{(i)}_{s\to t}$, $X = \mathbf{x}_s$ and $B = -\alpha^{(i)}_s$, one can easily obtain equation 5 to downdate the covariance matrices.

A.2 Downdating Covariance Matrices

From equation 23 we can write:

$$\begin{aligned}
\mathrm{R}^{(i)}_{(s+1)\to t}\Theta^{(i)}_{(s+1)\to t} &= \mathrm{X}^\top_{(s+1)\to t}\mathrm{A}^{(i)}_{(s+1)\to t}\mathrm{Y}_{(s+1)\to t} \\
&= \mathrm{X}^\top_{s\to t}\mathrm{A}^{(i)}_{s\to t}\mathrm{Y}_{s\to t} - \mathbf{x}_s\alpha^{(i)}_s\mathbf{y}_s \\
&= \mathrm{R}^{(i)}_{s\to t}\Theta^{(i)}_{s\to t} - \mathbf{x}_s\alpha^{(i)}_s\mathbf{y}_s \\
&= (\mathrm{R}^{(i)}_{(s+1)\to t} + \mathbf{x}_s\alpha^{(i)}_s\mathbf{x}_s^\top)\Theta^{(i)}_{s\to t} - \mathbf{x}_s\alpha^{(i)}_s\mathbf{y}_s \\
&= \mathrm{R}^{(i)}_{(s+1)\to t}\Theta^{(i)}_{s\to t} + \mathbf{x}_s\alpha^{(i)}_s\mathbf{x}_s^\top\Theta^{(i)}_{s\to t} - \mathbf{x}_s\alpha^{(i)}_s\mathbf{y}_s \\
&= \mathrm{R}^{(i)}_{(s+1)\to t}\Theta^{(i)}_{s\to t} - \mathbf{x}_s\alpha^{(i)}_s(\mathbf{y}_s - \mathbf{x}_s^\top\Theta^{(i)}_{s\to t})
\end{aligned}$$

which yield equation 18.

References

1. Almaksour, A., Anquetil, E.: Improving premise structure in evolving takagi-sugeno neuro-fuzzy classifiers. Evolving Systems 2, 25–33 (2011)
2. Angelov, P., Zhou, X.: Evolving fuzzy-rule-based classifiers from data streams. Fuzzy Systems, IEEE Transactions on 16(6), 1462 –1475 (2008)
3. Angelov, P., Filev, D.: An approach to online identification of takagi-sugeno fuzzy models. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 34(1), 484 – 498 (2004)
4. Delaye, A., Anquetil, E.: Hbf49 feature set: A first unified baseline for online symbol recognition. Pattern Recognition 46(1), 117 – 130 (2013)
5. Fortescue, T., Kershenbaum, L., Ydstie, B.: Implementation of self-tuning regulators with variable forgetting factors. Automatica 17(6), 831–835 (1981)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
7. Gama, J., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 329–Ŭ338 (2009)
8. Hägglund, T.: New estimation techniques for adaptive control (1983)
9. Haykin, S.O.: Adaptive Filter Theory (4th Edition). Prentice Hall, 4 edn. (2001)
10. Kulhavy, R., Zarrop, M.: On a general concept of forgetting. International Journal of Control 58(4), 905–924 (1993)
11. Liavas, A., Regalia, P.: On the numerical stability and accuracy of the conventional recursive least squares algorithm. Trans. Signal Processing 47(1), 88–96 (1999)
12. Lughofer, E.: Evolving fuzzy models: incremental learning, interpretability, and stability issues, applications. VDM Verlag Dr. Müller (2008)
13. Renau-Ferrer, N., Li, P., Delaye, A., Anquetil, E.: The ILGDB database of realistic pen-based gestural commands. International Conference on Pattern Recognition (ICPR 2012) (November 2012), tsukuba (Japan)
14. Salgado, M.E., Goodwin, G.C., Middleton, R.H.: Modified least squares algorithm incorporating exponential resetting and forgetting. International Journal of Control 47(2), 477–491 (1988)
15. Takagi, T., Sugeno, M.: Fuzzy Identification of Systems and Its Applications to Modeling and Control. IEEE Transactions on Systems, Man, and Cybernetics 15(1), 116–132 (1985)
16. Viard-Gaudin, C., Lallican, P.M., Binter, P., Knerr, S.: The ireste on/off (ironoff) dual handwriting database. In: Proceedings of the Fifth International Conference on Document Analysis and Recognition. pp. 455–. ICDAR '99 (1999)
17. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)