Last DRAFT
Electronic version of an article published as International Journal of Pattern Recognition and Artificial
Intelligence Vol 21, N°1, pp 99-116, DOI : 10.1142/S0218001407005326


# Writer Style Adaptation in On-line Handwriting Recognizers by a Fuzzy Mechanism Approach : The ADAPT Method

HAROLD MOUCHÈRE
ÉRIC ANQUETIL
NICOLAS RAGOT
*IRISA / CNRS / INSA de Rennes,*
*Campus Universitaire de Beaulieu, Avenue du Général Leclerc*
*Rennes, 35042, France*
*] {Harold.Mouchere, Eric.Anquetil} @irisa.fr*
*nicolas.ragot@univ-tours.fr*
*http://www.irisa.fr/imadoc*

This study presents an automatic on-line adaptation mechanism to the handwriting style of a writer for the recognition of isolated handwritten characters. The classifier we use here is based on a Fuzzy Inference System (FIS) similar to those we have designed for handwriting recognition. In this FIS each premise rule is composed of a fuzzy prototype which represents intrinsic properties of a class. Furthermore, the conclusion part of rules associates a score to the prototype for each class. The adaptation mechanism affects both the conclusions of the rules and the fuzzy prototypes by re-centering and re-shaping them thanks to a new approach called ADAPT inspired by the Learning Vector Quantization. Thus the FIS is automatically fitted to the handwriting style of the writer that currently uses the system. Our adaptation mechanism is compared with well known adaptation techniques. The tests were based on eight different writers and the results illustrate the benefits of the method in term of error rate reduction (86% in average). This allows such kind of simple classifiers to achieve up to 98.4% of recognition accuracy on the 26 Latin letters in a writer dependent context.

*Keywords*: supervised adaptation; on-line handwritten character recognition; Fuzzy Inference System.

## 1. Introduction

With the emergence of Personal Digital Assistants (PDA) and smartphones using pen-based interfaces, the handwriting recognition accuracy becomes very important, in terms of high recognition rates and low resource costs. Even if writer independent recognizers are more and more accurate for unconstrained handwriting, they remain error-prone. In fact, in the context of real world applications, they have to deal with many different writing styles, and the error rate is still too high for many users. One solution to overcome this limitation is to design adaptation techniques to optimize a writer independent system by using writer dependent specialization. This adaptation to the writer's style specificities must be fast, transparent and easy

for the user. The difficulty is then to learn quickly a new writer style with very few data and resources available.

There are several ways to adapt a classifier and they depend mainly on two things: the classifier kind and the available data for the adaptation process. A first way is off-line adaptation with an existing database of the user's handwriting to re-train the classifier, as in Ref. (8) with Hidden Markov Models (HMM). In this study, we focus on a second way, an incremental on-line adaptation that can be performed on devices with low resources by using only the last new characters inputted by the user. This was performed for example in Ref. (6) with HMM which are re-trained after few words inputted by the user. Closer from ours approach, works in Ref. (14) and Ref. (19) use K-nearest neighbor systems and they adapt the recognition system at each new character inputted.

In previous works, we have already designed powerful recognition systems: Mélidis[15] which is a generic pattern recognition approach and RESIFCar[3], a recognition system dedicated to isolated handwritten characters. These systems are based on compact and robust Fuzzy Inference System (FIS),[2] which allowed us to embed RESIFCar on mobile phones marketed in Europe.[1]

In order to improve the performance of these systems, we present a new on-line adaptation mechanism to the handwriting style of the current writer. This adaptation is done automatically and progressively during the use of the system. In this study we focus on the problem of the on-line adaptation of simple FIS. The aim is to apply later on this mechanism to more complex and more powerful systems such as RESIFCar or Mélidis. In these FIS, the rules use in premise fuzzy prototypes which describe the classes of characters. The numeric conclusions weight the participation of the prototypes to each class.

FIS optimization techniques already exist as described in Ref. (12). For the optimization of the numeric conclusions of the rules, methods based on the least squares are often used, like the pseudo inverse method or the gradient descent method. For modification of the rules premises the main classical approaches are based on gradient descent learning or genetic algorithms. In the handwriting recognition field there are already some adaptation methods.[13,14,19] Among those, works about the adaptation of systems based on prototypes[14,19] are particularly interesting for FIS.

We present in this paper a new writer adaptation method and strategy. It is inspired from the Learning Vector Quantization[10] (LVQ) and Elliptical Fuzzy Competitive Learning[9] (EFCL). The adaptation strategy is designed to respect the constraints imposed by the application frame *i.e.* on the one hand the incremental on-line adaptation progressively done all along the use and the stability of the performances in time and on the other hand the availability of few resources as in smartphones.

The paper is structured as follows. Firstly, the section 2 presents the properties of the used fuzzy inference system. Next, section 3 describes the adaptation ap-

proach by focusing on its originality compared to other existing techniques. Then, section 4 reports experimental results and comparisons on several writers for isolated handwritten character recognition. Finally, some perspectives and extensions are drawn in the conclusion.

## 2. Principles of the FIS

### 2.1. *Description of the FIS used*

The used classifier is formalized by an order zero Takagi-Sugeno FIS[17] with $N$ rules. A rule is composed of a premise (the *if-part*) and a conclusion (the *then-part*). FIS make a link with fuzzy rules between intrinsic models describing the properties of the handwritten characters and the corresponding label.[2] Each intrinsic model is defined by a set of fuzzy prototypes $P_r$ in $n$ dimensions. In the case of a $K$ classes problem, for each fuzzy prototype $P_r$, a rule $R_r$ is built:

$$\textbf{IF } \vec{X} \text{ is } P_r \textbf{ THEN } s_1^r = a_1^r \textbf{ and ... and } s_c^r = a_c^r \textbf{ and ... and } s_K^r = a_K^r,$$

where $\vec{X}$ is the feature vector of the character $X$ to recognize. As each prototype can take part in the description of each class, the rule $R_r$ has numeric conclusions which connect the prototype with each class $C$ by a prototype score $s_c^r$. The $a_c^r$ values are the weights corresponding to the participation of each prototype in the description of each class.

### 2.2. *Learning phase*

Initially the system is automatically trained from a learning database. The fuzzy prototypes are learned *separately on each class* thanks to an unsupervised clustering algorithm based on the possibilistic C-means.[11] Thus, the prototypes represent an intrinsic description of the classes.[15] The fuzzy prototypes $P_r$ are defined by their membership degree $\beta_r(\vec{X})$ of Eq. (1). This degree is an hyper-ellipsoidal radial basis function of center $\vec{\mu_r}$ and its shape is given by a covariance matrix $Q_r$ using the Mahalanobis distance[11] $d_{Q_r}(\vec{X}, \vec{\mu_r})$ :

$$\beta_r(\vec{X}) = \frac{1}{1 + d_{Q_r}(\vec{X}, \vec{\mu_r})}. \tag{1}$$

The conclusions $a_c^r$ of each rule are computed with the pseudo-inverse method.[4] This gives the optimum values to discriminate between classes by solving a linear equation system.

### 2.3. *Recognition process*

To determine the class of an unknown character $X$, its membership degrees $\beta_r$ to the $N$ fuzzy prototypes are computed according to Eq. (1) and the *sum-product*

inference is used to compute class scores $s_c$, *i.e.* the system outputs, with Eq. (2).

$$s_c = \frac{\sum_{r=1}^{N} \beta_r s_c^r}{\sum_{r=1}^{N} \beta_r}. \quad (2)$$

This equation shows how the different prototypes participate to the recognition of all classes.

The classification decision is then carried out by choosing the class $C_d$ which has the best (maximum) class score among the $s_c$ (Eq. (3)).

$$s_{C_d} = \max_c s_c. \quad (3)$$

## 3. On-line adaptation principles

The structure and the learning process of the used FIS make it quite similar to prototype based recognition approaches such as K-nearest neighbor classifiers. It is why the adaptation process proposed here is mainly inspired by the adaptation mechanism of K-nearest neighbor classifiers[14,19] *i.e.* the LVQ principle. The difference is that our FIS use hyper-ellipsoidal radial basis functions and numeric conclusions. So, our approach is also guided by the Elliptical Fuzzy Competitive Learning,[9] by the FIS[12] learning and by the radial basis function classifiers learning process.[16]

The writer adaptation is done during the use of the system and must respect the embedded constraints. The presented approach is thus iterative, *i.e.* it uses only the last current example (the character that has been just written by the user) or a buffer containing the last examples to adapt the system. Furthermore, the adaptation is supervised: each example is correctly labeled. This labeling is possible by asking the user to check the recognition or by using an auto-supervised technique like the one used in Ref. (14)

In the used FIS, the adaptation can be made in different ways in order to better discriminate the classes. First the prototypes used in *if-parts* (premises) can be re-centered, re-shaped, removed and it is also possible to add new ones to take into account the specificities of the writer. Secondly the conclusions of the *then-parts* can also be optimized in order to re-estimate the participation of each prototype to each class. We define an *adaptation cycle* as a sequence consisting in a *premise adaptation* and then a *conclusion adaptation* for one example (character).

In this study we focus on how to adapt the premises of the system rules by re-centering and re-shaping prototypes. The addition and the removal of prototypes will be studied in future works.

In the following section 3.1, we present in more details the originality of the approach used to adapt the premises by re-centering and re-shaping the prototypes. The section 3.2 presents the classical Gradient Descent method used to adapt the conclusions. After that, we present in section 3.3 how to use and combine these two adaptation steps in an embedded application.

### 3.1. *Premises adaptation*

For both the re-centering and the re-shaping we present firstly the direct transposition of existing approaches and secondly our ADAPT approach.

#### 3.1.1. *Prototype re-centering approaches*

Re-centering the prototypes makes it possible to better represent the specificity of the writing style of a new writer. There already exists some unsupervised techniques which re-center prototypes like Competitive Learning (CL) or Fuzzy Competitive Learning (FCL).[9] But as we focus on supervised techniques, the used process is instead inspired by the Learning Vector Quantization (LVQ) algorithm.[10] Different versions of this algorithm have been compared in Ref. (19) to adapt K-nearest neighbor classifiers. The simplest supervised version (LVQ1) brings the nearest prototype closer to the example if it is correctly classified and moves the prototype away from the example if it is misclassified. This technique supposes that the prototypes are crisply labeled. The direct transposition of this method to our FIS could be proceeded in the following way: during the learning process we have labeled each prototype with the class from which it was learned and during the adaptation process the most activated prototype is re-centered according to its original class (winner-take-all method). The center $\vec{\mu_r}$ of the prototype $P_r$ of the rule $r$ is updated with the displacement vector $\Delta\vec{\mu}_r$:

$$\Delta\vec{\mu}_r = \lambda * \delta * (\vec{X} - \vec{\mu_r}), \tag{4}$$

with $\delta$ set to 1 if $X$ has the same class than $P_r$ and to -1 otherwise. The adaptation parameter $\lambda$ lies between 0 and 1. It controls the amplitude of the displacement and thus the adaptation rate. The value of $\lambda$ is discuss in section 3.3. If this method is used directly on our FIS, there will be no good results as shown in section 4. The reason is that our prototypes are not crisply labeled as they participate in the recognition of all classes (*cf.* section 2.3) since in a FIS all prototypes are taken into account to recognize an entry, the decision does not depend only on one prototype.

An extension of this mechanism consists in re-centering all the prototypes according to the activation of the corresponding premise unlike winner-take-all methods. This method is transposed from the Fuzzy Learning Vector Quantization (FLVQ) used for a supervised fuzzy competitive learning based on prototypes.[7] The farther the activation $\beta_r$ of the premise $r$ is away from its objective score $\beta_r^*$, the more it should be moved:

$$\Delta\vec{\mu}_r = \lambda * (\beta_r^* - \frac{\beta_r}{\sum_{q=1}^{N} \beta_q}) * (\vec{X} - \vec{\mu_r}). \tag{5}$$

The objective score $\beta_r^*$ is 1 if the prototype $P_r$ and $X$ belong to the same class and 0 otherwise. FLVQ uses the activation of each prototype for the center update. Thus, at each adaptation cycle all of them can be modified. But they are still labeled crisply and do not take into account all information used in FIS. For example,

a prototype from another class will be moved away even if it participates in the recognition process *via* conclusion scores. Furthermore, if more than one prototype are used for the class description, FLVQ tends to improve the activation of a far away prototype, even if another one permits the classification.

#### 3.1.2. *The ADAPT prototype re-centering approach*

The transposition of LVQ1 and FLVQ that we use comes from K-nearest neighbor (fuzzy) classifiers. Thus, they need crisply labeled prototypes and do not take into account the participation of each prototype in the final class score. In this context, a beneficial displacement for a class can be wrong for other classes. This phenomenon limits the adaptation and can have bad effects for some writers (*cf.* section 4.2).

Thus, we propose the ***AD****aptation by* ***A****djustment of* ***P****roto****T****ypes* (ADAPT) method. It allows to modify all the prototypes of the FIS by re-centering them for each new example that is inputted. This is done according to their participation in the recognition process. The prototypes are not labeled and thus participate in the description of all the classes. The update of prototype must improve the score of each class. In this way, the displacement $\Delta\vec{\mu}_r$ must be significant if the class score $s_c$ is different from those wanted, the participation $s_c^r$ of the prototype to the final decision is high and the rule premise is activated. Equation (6) gives the prototype update using the proposed ADAPT learning rate $\delta_r^{'}$:

$$\Delta\vec{\mu}_r = \lambda\delta_r^{'}(\vec{X} - \vec{\mu_r}) \tag{6}$$

$$\delta_r^{'} = \beta_r \sum_{c=1}^{C} \left((b_c - s_c)\, s_c^r\right), \tag{7}$$

with $b_c$ the wanted class score for $s_c$ : 1 if $c$ is the example class and 0 otherwise.

The Figure 1 shows an example of this compromise. The prototype was learned initially on the class 1 and an example of this class is presented to adapt the system. The first idea is to move the prototype closer to the example but, as this prototype participate to the recognition of other classes (class 2 and 3 in this example) this move can have a bad effect on the recognition. So we also consider the re-centering needed to optimize the class scores of these two classes. The final re-centering is the sum vector of all the needed moves.

Thus, we can rewrite the equation Eq. (6) as a sum of displacements where each displacement improves to 1 the class score of the class of the example and decreases to 0 for the other classes:

$$\Delta\vec{\mu}_r = \lambda \sum_{c=1}^{C} \left(\beta_r s_c^r (b_c - s_c)(\vec{X} - \vec{\mu_r})\right). \tag{8}$$

The ADAPT prototype update is thus a compromise between the improvements of each class score.

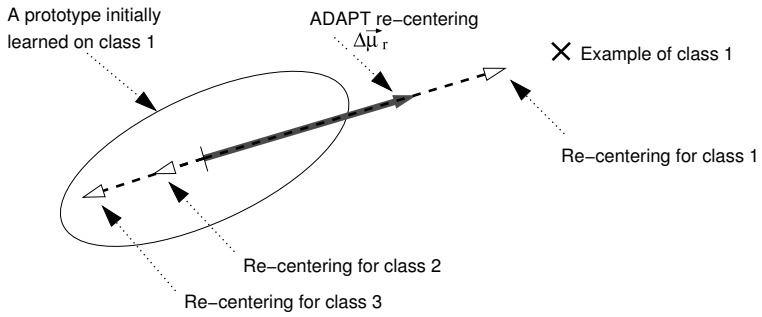



Fig. 1. Principle of the ADAPT compromises between the optimizations of all classes.

3.1.3. *Prototype re-shaping approaches*

The re-centering of the prototypes allows to fit the new localization of the writer data in the input space. To better represent the repartition of these new data, the shape of the prototypes must also be adapted. The shape of the prototype $P_r$ is given by its associated covariance matrix $Q_r$. So, re-shaping the prototypes corresponds to the re-evaluation of these matrices. Nevertheless, the Mahalanobis distance uses the inverse matrix $Q_r^{-1}$, so it is more efficient to update directly the inverse matrix than re-evaluating the covariance matrix first and then inversing it at each adaptation cycle.

An iterative formula is given by Schürmann[16] to recursive estimation of a covariance matrix in an unsupervised context:

$$Q_r \Leftarrow (1-\alpha)\left(Q_r + \alpha(\vec{X}-\vec{\mu_r})(\vec{X}-\vec{\mu_r})^T\right), \tag{9}$$

with parameter $\alpha$ the learning rate. This formula can be transformed to estimate the inverse covariance matrix as shown in Ref. (16):

$$Q_r^{-1} \Leftarrow \frac{Q_r^{-1}}{1-\alpha} - \frac{\alpha}{(1-\alpha)}\frac{(Q_r^{-1}\vec{m})(Q_r^{-1}\vec{m})^T}{1+\alpha(\vec{m}^T Q_r^{-1}\vec{m})}, \tag{10}$$

with $\vec{m} = \vec{X} - \vec{\mu_r}$ and $\alpha$ is the learning rate.

3.1.4. *The ADAPT prototype re-shaping approach*

The drawback of the previous update method is that it is unsupervised and it can not take into account neither the numeric conclusions of the FIS nor the error on each class. Consequently, as in Ref. (9) where EFCL uses the activation of the prototype, we propose to replace $\alpha$ in Eq. (9) by $\alpha\delta_r^{'}$ which uses the ADAPT learning rate $\delta_r^{'}$ from Eq. (7). The value of $\alpha$ is discuss in section 3.3. We can thus rewrite the equation Eq. (10) to estimate the inverse matrix using supervised

information:

$$Q_r^{-1} \Leftarrow \frac{Q_r^{-1}}{1-\alpha\delta_r'} - \frac{\alpha\delta_r^{'}}{1-\alpha\delta_r'} \cdot \frac{(Q_r^{-1}\vec{m}) \cdot (Q_r^{-1}\vec{m})^T}{1+\alpha\delta_r'(\vec{m}^T Q_r^{-1}\vec{m})} \ . \tag{11}$$

By this way, the ADAPT re-shaping, as the ADAPT re-centering, uses the activation of the prototype, its participation on the recognition of each class and the error made on each class.

### 3.2. *Conclusion adaptation*

To provide the conclusion adaptation, the classical Gradient Descent (GD) method is chosen because it is simple, requires few resources and can be used in an iterative way. This method (Eq. (12)) updates the numeric conclusions of the rules considering the prototype scores, the class scores and the target class score $b_c$. This target class score $b_c$ is 1 if $c$ is the class of the example and 0 otherwise. The adaptation parameter $n$ lies between 0 and 1 and controls the adaptation speed.

$$\Delta s_c^r = n * (b_c - s_c) * \beta_r. \tag{12}$$

### 3.3. *On-line adaptation strategies*

The aim of adaptation strategies is to obtain a fast and robust adaptation with respect to the constraints of an on-line adaptation process embedded in a small device such as a smartphone.

A robust adaptation could be obtained by storing all the previous examples inputted by the user and then adapting the system to them. But, here it is impossible because of the limitation of the memory resources. Thus, in order to have some diversity in the examples and to increase the adaptation speed, the last $F$ examples are stored in a data buffer. Each time a new example is inputted, it is added to the data buffer and the oldest one is removed. An *adaptation cycle*, as defined in section 3.1, is run for each example stored in this buffer. So $F$ is an adaptation parameter which influences the computing time.

An other way to increase the speed and robustness of the adaptation consists in defining the value of the learning rates $\lambda$ (Eq. (6)) and $\alpha$ (Eq. (11)). Whereas, a high value allows a fast but unstable adaptation, a low one allows a stable and robust but slower adaptation. So we use a classical[7,10,16] mechanism which decreases the learning rates. Thus, a decreasing learning rate allows a fast and robust adaptation. The original aim of this technique is to allow the same importance to all examples used in this learning process (the adaptation for us). Nevertheless, in our context of adaptation we use a decreasing half bell shape curve to limit the decreasing at the beginning and to have a minimum value in order to keep an adaptation even after a long use.

The decreasing from $\lambda_{max}$ to $\lambda_{min}$ is defined here by:

$$\lambda(t) = \frac{\lambda_{max} - \lambda_{min}}{1 + t/T} + \lambda_{min}, \tag{13}$$

where $t$ is incremented for each new character inputted by the user, after the adaptation on the data-window. When $t = T$ we have $\lambda = \frac{1}{2}(\lambda_{max} + \lambda_{min})$.

The same technique is used for $\alpha$, the deformation parameter from Eq. (11):

$$\alpha(t) = \frac{\alpha_{max} - \alpha_{min}}{1 + t/T} + \alpha_{min}, \tag{14}$$

## 4. Experiments

In order to validate our approach we compare it with the direct transposition of the existing methods LVQ (Eq. (4)) and FLVQ (Eq. (5)) using a set of eight different writers. In this first study, we focus on the adaptation speed (in terms of number of inputted examples) and on the robustness (in terms of stability). After that we study the behavior of the ADAPT method in a simulation of a real use context.

### 4.1. *Experimental protocol*

The experiments are based on the recognition of the 26 lower case Latin letters, without any constraints for the writer. The initial learning of the system uses 5287 characters of the Ironoff database[18] which contains about 400 writers. The writer specific databases were written on a PDA by eight users all different from those involved in the Ironoff database. Each writer has inputted 40 times each characters *i.e.* 1040 characters per writer. In this experiment, there was no recognition feedback so the writer can not adapt his style to the recognition system. To estimate the adaptation performance on each writer, we proceed by a four-fold cross-validation technique. 3/4 of the writer database (780 letters) is used to adapt the system to him and 1/4 (260 letters) is used to evaluate the results of this adaptation to his personal handwriting style. To observe the adaptation effects during a longer use, the adaptation databases are used twice. For each split of the data, this adaptation is carried out five times with a different order for the adaptation data in order to avoid effects due to the letter input order. The presented curves and results are thus the average of these 20 tests (five times the four-fold cross validations).

In this experiment, the characters are described by a set of 21 features similar to those used in ResifCar.[1] The class description uses two prototypes. As there are 26 classes the system has 52 prototypes and so 52 rules.

### 4.2. *Global results*

#### 4.2.1. *Comparison of re-centering methods*

The table 1 shows the recognition rates before and after adaptation with the different methods for each writer. The last column shows the average recognition rate (ARR) on all writers' databases. "GD" alone is an adaptation with just the Gradient Descent method without any prototype updating. "X+GD" represents the adaptation with a complete adaptation cycle *i.e.* re-centering method X and GD.

The method X can be LVQ1 (Eq. (4)), FLVQ (Eq. (5)) or ADAPT (Eq. (6)). The GD parameter $n$ is 0.14, the parameter $\lambda$ decreases from 0.05 to 0.005 for LVQ1 and ADAPT and from 0.005 to 0.0005 for FLVQ and the data window size $F$ is 20. The value of $T$ for $\lambda$ decrease is 100. These values were found empirically in order to optimize the results.

Table 1. Recognition rates before and after adaptation with the different methods.

| | Writer | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Adaptation | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ARR |
| Before | 88.9 | 90.7 | 87.8 | 90.1 | 87.6 | 91.6 | 85.2 | 87.5 | 88.7 |
| GD | 92.3 | 92.5 | 91.0 | 93.3 | 92.7 | 92.0 | 87.6 | 92.2 | 91.7 |
| FLVQ+GD | 94.1 | 89.1 | 92.6 | 94.2 | 94.8 | 93.9 | 90.4 | 92.3 | 92.7 |
| LVQ1+GD | 95.8 | 93.5 | 93.2 | 95.9 | 95.3 | 94.1 | 93.2 | 95.3 | 94.5 |
| ADAPT+GD | 97.2 | 95.4 | 95.3 | 97.4 | 97.1 | 96.3 | 95.9 | 97.7 | 96.5 |
| ADAPT+ re-shaping+GD | 98.9 | 97.1 | 97.4 | 99.0 | 99.2 | 98.3 | 98.9 | 98.6 | 98.4 |

Firstly, we can see that the use of the prototype center update improves the adaptation results compared to the conclusion adaptation GD alone. Secondly, the ADAPT method achieves the best recognition rates: 96.5% in average *i.e.* an error reduction of 69% against an error reduction of 51% for LVQ1. The transposed FLVQ method achieves lower results than the transposed LVQ1 method. It shows that the transpositions of LVQ and FLVQ methods do not fit with the type of used FIS. Actually, they keep the initial association of the prototypes with classes even though the prototypes describe all classes and this association can change during adaptation cycles through the conclusion adaptation. The lower results of the FLVQ method (especially for writer 2) probably comes from the fact that FLVQ updates all prototypes regardless of their contribution to each class and the LVQ1 method takes fewer risks by updating only one prototype.

So the ADAPT method is a more appropriated re-centering formula for this recognition system where the prototypes participate in the recognition of all classes.

#### 4.2.2. *Contribution of the re-shaping*

The last line of table 1 shows the recognition rate after adaptation using the ADAPT re-centering (Eq. (6)) and re-shaping (Eq. (11)) methods. The value of $\alpha$ varies from 0.005 to 0.001 according to Eq. (14) with $T = 100$.

We can see that the re-shaping allows another reduction of the error rate by 54% with regards to the use of just the ADAPT re-centering and 86% with regards to the initial recognizer. So with this complete adaptation skill the recognition rate rises up to 98.4% which represents just one error for 60 characters recognized.

### 4.3. *Results analysis*

#### 4.3.1. *Comparison of the recognition rates evolution*

Figure 2 compares the evolution of the average recognition rate of all writers during the adaptation with the different methods.

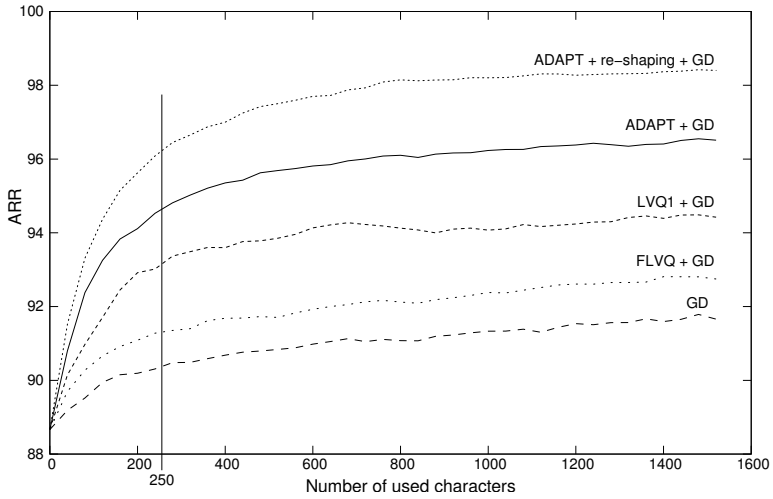



Fig. 2. Comparison of average recognition rate for different adaptation methods.

Firstly, our approach allows a stable adaptation *i.e.* the recognition rate does not fluctuate or decrease at the end. Secondly, we can see that the adaptation to the writer's style is faster with the ADAPT+GD method than with the others. For example after 250 characters ADAPT+GD achieves 94.5% whereas LVQ1+GD achieves 93%. We can notice that the re-shaping not only increases the final score but also increases the speed of the adaptation. Indeed, we adapt both the center and the shape of the prototypes and they are two complementary ways of improving the recognition. With only 250 characters (about 50 words) the recognition rate rises from 88.7% up to 96.2% using ADAPT+re-shaping+GD and it represents about 66% of the final adaptation. This adaptation speed is very interesting for our application context, where the user will rapidly have less errors to correct.

#### 4.3.2. *2D adaptation display*

In order to show the behavior of the ADAPT+GD method, we have done the same experiment with just three classes ("a", "f" and "x") and in two dimensions only (*i.e.* two features). These classes and features were chosen to have a suitable data repartition in the two dimension feature space with recognition rates comparable to those of the experiment in the initial 21 dimension space. Each class is initially

described by two prototypes *e.g.* Pa1 and Pa2 for the class "a". Thus, we can observe the different repartition of letters for each writer due to their personal handwriting style. We can also display the decision boundaries and the fuzzy prototypes as ellipses (the 0.5 $\alpha$-cut). Figure 3 shows a part of the learning database Ironoff, the initial six prototypes and the initial decision boundaries. Figures 4 and 5 show some examples of writers 4 and 8 respectively, the six prototypes and the decision boundaries after the adaptation ADAPT+GD with re-shaping.

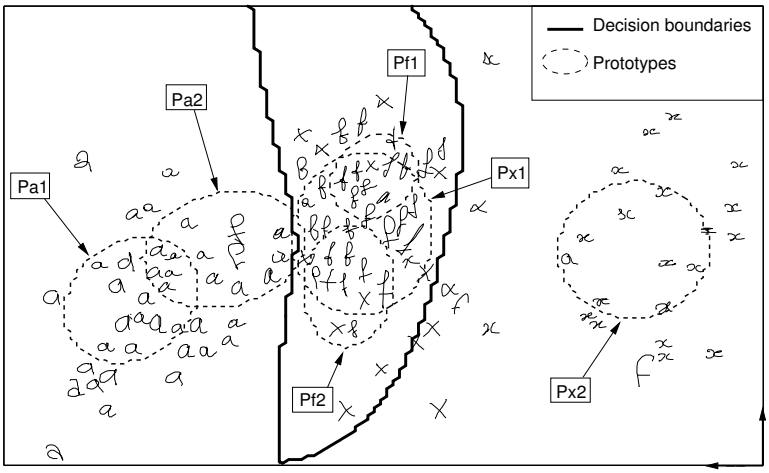



Fig. 3. The initial FIS for three classes in two dimensions with examples from the learning database. Pa1 and Pa2 are prototypes of "a". Pf1 and Pf2 are prototypes of "f". Px1 and Px2 are prototypes of "x".

We can see in Figure 4 that the writer 4 has a regular handwriting style. For example, he writes homogeneous "f" with two loops and cursive "x". Moreover, in Figure 3, the "x" and "f" classes have a great confusion which is not observed for the writer 4. After adaptation, the prototypes have been re-centered. For example, Pa2 has been centered at the new data location. Since Pa1, Pf1 and Px2 had already a good place, they moved only slightly. Furthermore, a prototype has changed the class that it represents (*cf.* Figure 3): Px1 describes now the class "f" instead of the class "x". Moreover, Pf2 seems to be unused by this user's writing style. In future works, these redundant or unused prototypes could be deleted.

Figure 5 shows the adaptation to writer 8 whose writing style is very different from the one of writer 4. We had to note that he writes homogeneous "a" and "f" but has two kinds of "x". We can see on this figure that there are clearly two "x" locations and that the adaptation process has moved the Pf2 prototype to one location of "x". Furthermore, in Figure 5, the prototype Px1 is now placed on class "f" instead of the initial class "x".

Other re-centering methods like LVQ1 would not permit these exchanges of

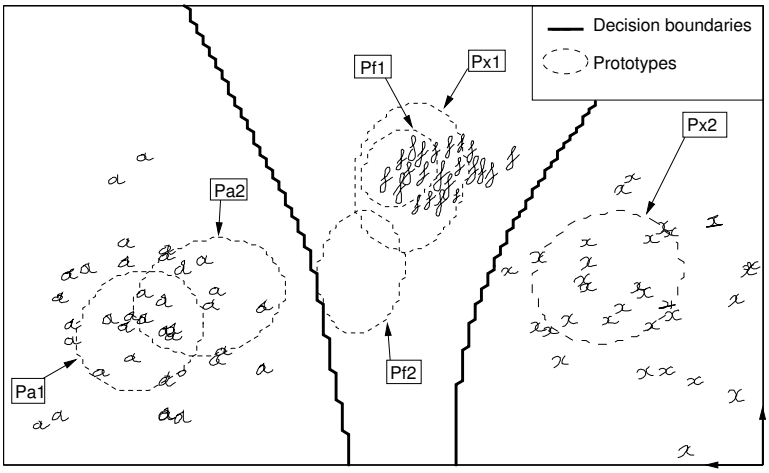



Fig. 4. The FIS for three classes in two dimensions after adaptation to writer 4.

prototypes (Px1 for writer 4 and Px1 and Pf2 for writer 8). Indeed, these methods use labeled prototypes and they would try to keep the labels even if the prototypes need to cross the feature space to be closer to their associated class.

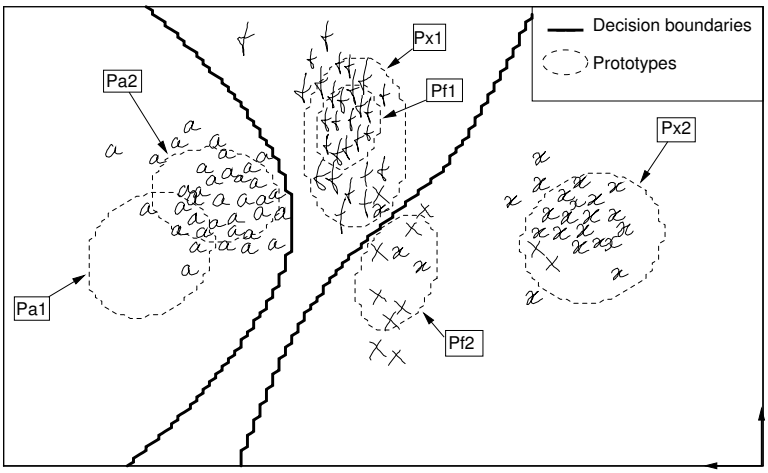



Fig. 5. The FIS for three classes in two dimensions after adaptation to writer 8.

Figure 6 shows the six prototypes before (from Figure 3) and after (from Figure 5) adaptation to writer 8 in order to appreciate the re-centering and the reshaping of each one. For example, Px1 has changed its height and width, but Pa2 and Pf2 have also changed their orientation. We can also see how far away were the

initial prototypes from the writer 8 style.

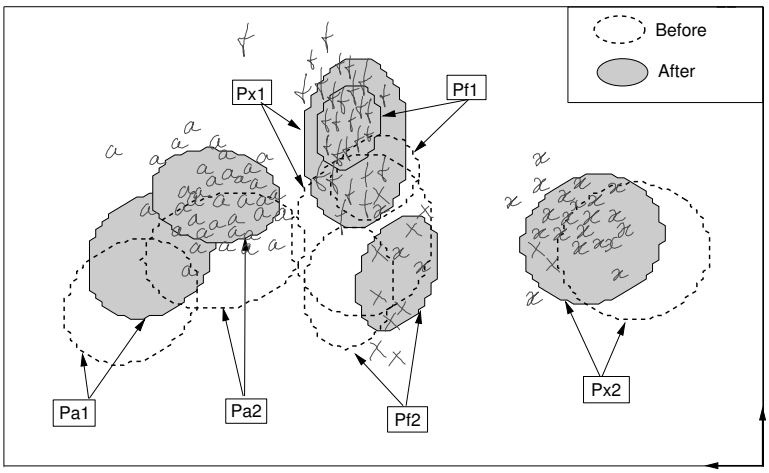



Fig. 6. Positions and shapes of prototypes before and after adaptation to writer 8.

In conclusion, the decision boundaries after the adaptation better discriminate all the classes for these writers. Indeed, their shapes and positions have been well adapted to the different handwriting styles.

#### 4.3.3. *Style adaptation*

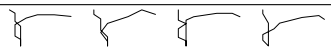
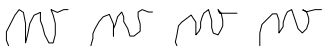
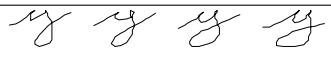
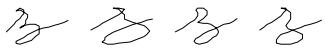
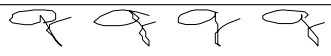
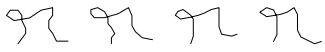
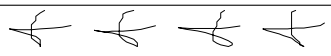
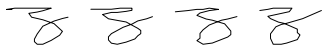
To observe the style adaptation, we compare the recognition results of the classifier before and after the adaptation, for each writer, by showing examples of writing styles which are misclassified by the original FIS and which are correctly classified after adaptation (ADAPT+GD). Table 2 reports these results and the error rates of the corresponding class before and after the adaptation. It also reports the relative variation $\Delta$ of the recognition error rate.

All these handwriting styles are probably represented in the learning database because of the number of writers in Ironoff. But there are some confusion errors between characters from different handwriting styles (for example, all errors in the class "q" of writer 7 are confusions with the class "g"). So the improvement (up to 99% for the class "q" of this writer) comes from a well adapted representation of the writer style specificity.

### 4.4. *Real experiment simulation*

In previous sections we have shown that our ADAPT strategy is able to fit the recognizer to the handwriting style of a writer. But in these experiments, all data are used in a random order with an equiprobable apparition for each class. In

Table 2. Examples of writing styles which are misclassified before adaptation and are correctly classified after.

| Writer | Class | Error rate (%) Bef. | Aft. | Δ(%) | Examples |
|---|---|---|---|---|---|
| 1 | r | 80 | 13 | -84 |  |
| | w | 55 | 0 | -100 |  |
| 4 | y | 53 | 3 | -95 |  |
| | z | 58 | 19 | -68 |  |
| 7 | q | 93 | 1 | -99 |  |
| | r | 68 | 5 | -93 |  |
| 8 | f | 93 | 3 | -97 |  |
| | z | 58 | 0 | -100 |  |

a real life experiment the characters arrive in the text order and with different probabilities. For example, the class "e" will appear more frequently than other classes and the trigram "the" is more probable than the trigram "ztw". So it could be interesting to know how effective ADAPT is in a more real context as described in Ref. (5): the user inputs handwritten characters one by one and the system recognizes them separately.

We consider a piece of a lowercase text write in table 3 and it is split into two parts. The first two sentences (266 characters) are used as an adaptation sequence and the next two sentences (237 characters) are used as a test sequence. The on-line characters of this two sequences are taken randomly in the adaptation and test databases respectively for each writer. Thus, the characters used in the adaptation sequence are not include in the test sequence.

Table 4 shows the test sentences as recognized by the initial FIS for writer 1 (with the correct character below each mistake). Table 5 shows the same sentences but after adaptation to writer 1. We can see that the system starts with 25 errors, which represents an initial recognition rate of 89.4% but the sentences are unreadable. After adaptation, the system makes only four errors (98.3% of recognition rate), which represents only one mistake every ten words. In a real context, a dictionary could be used to correct these remaining mistakes and this work will be much easier with this adaptation.

The same test is made with the four-fold cross-validation for all users and five draws of characters. In average, the recognition rate rises from 88.9% to 97.8% with 266 adaptation characters. It is better than the rate of the first experiments with

Table 3. The sentences used to adapt and test the FIS. Spaces and punctuations are ignored during the process.

| |
|---|
| in this study we present an automatic on-line adaptation mechanism to the writer's handwriting style for the recognition of isolated handwritten characters. the classifier we use here is based on a fuzzy inference system (fis) similar to those we have developed for handwriting recognition but simplified for this study. |
| doing so, the adaptation mechanisms presented here can be transposed to the original systems. in this fis each premise rule is composed of a fuzzy prototype which represents intrinsic properties of a class. the consequent part of rules associates a score to the prototype for each class. |

Table 4. The test sentences and the corrected errors before adaptation to writer 1 (25 errors *i.e.* 10.6% of the sentence).

| |
|---|
| doing sv the adaptatiom mechdnisms presented hbre cdn<br>o n a e a |
| be trdmsposed to the origindl systems in tkis fis bach<br>an a h e |
| premise rule is composed of a fuzry prototgpb whick represents<br>z y e h |
| intrinsic propertibs of a class the consequemt pdrt vf rubes<br>e n a o l |
| dssocidtbs d scorc to the prototgpe for each class<br>a a e a e y |

the same number of used characters which was 96.3%. This difference must be due to a good adaptation to frequent characters which are also frequent in the test sequence. It shows that our adaptation method will be able to reduce the number of errors in a real context even more rapidly and efficiently.

## 5. Conclusion

In the context of fuzzy classifier adaptation for on-line handwritten character recognition, we have presented a new adaptation approach, namely ADAPT. This approach is able to adapt prototype-based Fuzzy Inference System with numeric conclusions. This incremental adaptation is performed conjointly by re-centering

Table 5. The test sentences and the corrected errors after adaptation to writer 1 (4 errors *i.e.* 1.7% of the sentence).

| |
|---|
| doing so the adaptatiom mechanisms presented here can<br>n |
| be tramsposed to the original systems in this fis each<br>n |
| premise rule is composed of a fuzry prototype which represents<br>z |
| intrinsic properties of a class the consequemt part of rules<br>n |
| associates a score to the prototype for each class |

and re-shaping fuzzy prototypes and by re-evaluating the numeric conclusions. To achieve our aim we design a suitable on-line adaptation strategy which allows the adaptation to be quick and robust and to respect the embedding constraints.

The reported experiments show results for eight different writers. The ADAPT method used on prototype-based Fuzzy Inference System allows a better adaptation than the classical methods LVQ1 and FLVQ. Indeed, we obtain an high error reduction of 86% in average. The recognition rate rises from 88.7% to 98.4% in average (up to 99.2% for the best writer). Another experiment close to a real context of use illustrates how this adaptation strategy is able to reduce the number of errors from one character every two words to one character every ten words.

The method will be extended in future works on the one hand to remove unused or redundant prototypes to simplify the recognizer and on the other hand to add prototypes to have a more efficient adaptation in terms of accuracy and rapidity. Furthermore, the next step will be to apply these adaptation methods to the more complex system of handwriting recognition RESIFCar.

## Acknowledgments

Authors would like to thank Guy Lorette, Professor at the University of Rennes 1, for his precious advice. This work is supported by the CNRS (Centre National de la Recherche Scientifique) and the Brittany Region.

## References

1. E. Anquetil and H. Bouchereau, Integration of an on-line handwriting recognition system in a smart phone device, *Proc. 16th Int. Conf. on Pattern Recognition*, Vol. 3, Quebec, Canada, 2002, pp. 192–195.
2. E. Anquetil and G. Lorette, Automatic generation of hierarchical fuzzy classification systems based on explicit fuzzy rules deduced from possibilistic clustering: Application to on-line handwritten character recognition, *Proc. 6th Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Grenade, Spain, 1996, pp. 259–264.
3. E. Anquetil and G. Lorette, On-line handwriting character recognition system based on hierarchical qualitative fuzzy modeling, *Proc. 5th Int. Workshop on Frontiers in Handwriting Recognition*, Colchester, England, 1996, pp. 47–52.
4. C. Bishop, *Neural Network for Pattern Recognition*, Oxford University Press, 1995.
5. F. Bouteruche, G. Deconde, E. Anquetil, and E. Jamet, Design and evaluation of handwriting input interfaces for small-size mobile devices, *Proc. 1st Workshop on Improving and Assessing Pen-Based Input Techniques*, Edinburgh, Scotland, 2005, pp. 49–56.
6. A. Brakensiek, A. Kosmala, and G. Rigoll, Comparing adaptation techniques for on-line handwriting recognition, *Proc. 6th Int. Conf. on Document Analysis and Recognition*, Seattle, WA, USA, 2001, pp. 486–490.
7. F. Chung and T. Lee, Fuzzy competitive learning, *IEEE Trans. on Neural Network*, **7**(3) (1994) pp. 539–551.
8. S. Connell and A. Jain, Writer adaptation for online handwriting recognition, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24**(3) (2002) pp. 329–346.
9. S. De Backer and P. Scheunders, Texture segmentation by frequency-sensitive elliptical competitive learning, *Image and Vision Computing*, **19**(9–10) (2001) pp. 639–648.
10. T. Kohonen, The self-organizing map, *Proceeding of IEEE*, **78**(9) (1990) pp. 1464–1480.
11. R. Krishnapuram and J. Keller, A possibilistic approach to clustering, *IEEE Trans. on Fuzzy Systems*, **1**(2) (1993) pp. 98–110.
12. B. Mitaim, S. Kosko, The shape of fuzzy sets in adaptive function approximation, *IEEE Trans. on Fuzzy Systems*, **9**(4) (2001) pp. 637–656.
13. A. Nakamura, A method to accelerate adaptation for on-line handwriting recognition of large character set, *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*,Tokyo, Japan, 2004, pp. 426–431.
14. L. Oudot, L. Prevost, A. Moises, and M. Milgram, Self-supervised writer adaptation using perceptive concepts : Application to on-line text recognition, *Proc. 17th Int. Conf. on Pattern Recognition*, Vol. 2, Cambridge, UK, 2004, pp. 598–601.
15. N. Ragot and E. Anquetil, Melidis: Pattern recognition by intrinsic/discriminant dual modeling based on a hierarchical organization of fuzzy inference systems, *Proc. 10th Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Perugia, Italy, 2004, pp. 2069–2076.
16. J. Schürmann, *Pattern Classification: a unified view of statistical and neural approaches*, Wiley-Interscience, 1996.
17. T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. on Systems, Man and Cybernetics.* **15**(1) (1985) pp. 116–132.
18. C. Viard-Gaudin, P. Lallican, S. Knerr and P. Binter, The irest on/off dual handwriting database, *Proc. 5th Int. Conf. on Document Analysis and Recognition*, Bangalore, India, 1999, pp. 455–458.

19. V. Vuori, J. Laaksonen and E. Oja, On-line adaptation in recognition of handwritten alphanumeric characters, *Proc. 5th Int. Conf. on Document Analysis and Recognition*, Bangalore, India, 1999, pp. 792–795.