On-line writer adaptation for handwriting recognition using fuzzy inference systems

Harold Mouchère, Eric Anquetil, Nicolas Ragot

On-line Writer Adaptation for Handwriting Recognition using Fuzzy Inference Systems

Harold Mouchère* Eric Anquetil Nicolas Ragot

IRISA, INSA de Rennes
Campus Universitaire de Beaulieu, Avenue du Général Leclerc
35042 Rennes Cedex, France
{*Harold.Mouchere, Eric.Anquetil, Nicolas.Ragot*}*@irisa.fr*

Abstract

We present an automatic on-line adaptation mechanism to the writer's handwriting style for the recognition of isolated handwritten characters. The classifier is based on a Fuzzy Inference System (FIS). This FIS is composed of fuzzy prototypes which represent the intrinsic properties of the classes and it uses numeric conclusions. The proposed adaptation mechanism affects both the conclusions of the rules and the fuzzy prototypes of the premises by re-centering and re-shaping them. Doing so, the FIS is automatically fitted to the handwriting style of the writer that is currently using the system. This adaptation mechanism has been tested with 8 different writers. The results show the adaptation mechanism is able to improve the recognition rate from 88% to 98.2% in average for the 26 Latin letters.

1. Introduction

With the emergence of Personal Digital Assistants (PDA) and of smartphones using pen based interfaces, the handwriting recognition accuracy is very important, in term of high recognition rates and of low resource costs. One way to improve the accuracy is to adapt the recognition system to the writer's style specificities. This adaptation must be fast, transparent and easy for the user. The difficulty is to learn quickly a new writer style with few data and few resources available.

There are several ways to adapt a classifier. It can be done off-line with an existing database, as in [6] with Hidden Markov Models (HMM). But only the on-line adaptation is interesting in our context because it just needs the new characters inputted by the user. This was performed for example in [5] with HMM and in [11, 16] with k-nearest neighbor systems.

In previous works, we have already designed two powerful recognition systems: Mélidis [12] is based on a generic pattern recognition approach and RESIFCar [3] is a system dedicated to the recognition of isolated handwritten characters. These systems are based on compact and robust Fuzzy Inference Systems (FIS) [2], which have allowed to embed RESIFCar on mobile phones [1]. In these FIS, the rules use fuzzy prototypes in premise which are describing each class of characters. The numeric conclusions weight the participation of the prototypes for each class.

Our aim is to provide an adaptation mechanism for such kind of FIS. More generally, the adaptation of a FIS can be seen as some optimization technique. In [10] optimizations of different FIS are presented. For the optimization of the numeric conclusions of the rules, methods based on the least squares are often used, as the pseudo inverse or the gradient descent. For the modification of the rules premises, the main classical approaches are based on gradient descent learning or genetic algorithms. But these techniques require lots of computing time and lots of available data.

We present in this paper a new writer adaptation method called ADAPT. It is inspired from the Learning Vector Quantization (LVQ) [8] and Elliptical Fuzzy Competitive Learning (EFCL) [7]. The adaptation strategy is designed to respect the constraints imposed by the application frame: on-line adaptation, i.e. progressively all along the use; few resources available as in smartphones; stability of the performances in time. We firstly experiment the proposed method on simple FIS. But the aim is to apply later this mechanism to our more complex and more powerful handwriting recognition systems.

The paper has the following organization. Firstly, the section 2 presents the properties of the used fuzzy inference system. Next, section 3 describes the adaptation approach by focusing on its originality. Then section 4 reports experimental results on several writers for isolated handwritten characters recognition.

* This work is supported by the Brittany Region.

Proceedings of the 2005 Eight International Conference on Document Analysis and Recognition (ICDAR'05)
1520-5263/05 $20.00 © 2005 IEEE

2. Principles of the used FIS

2.1. Description

The classifier is formalized by an order 0 Takagi-Sugeno FIS [14]. The fuzzy rules make a link between intrinsic models that describe the properties of the handwritten characters and the corresponding label [2]. Each intrinsic model is defined by a set of fuzzy prototypes P_i in n dimensions. For a K classes problem, a rule R_i is built for each P_i:

IF $\vec{X}$ is P_i **THEN** $s_1^i = a_{i1}$ **and ... and** $s_c^i = a_{ic}$ **and ... and** $s_K^i = a_{iK}$,

where $\vec{X}$ is the feature vector of the character X to recognize. As each prototype can participate to the description of each class, the rule R_i has numeric conclusions connecting P_i with each class $c = 1..K$ by a prototype score s_c^i. The a_{ic} is a weight that expresses the participation of P_i in the description of the class c.

2.2. Learning

Initially, the system is automatically trained with a learning database. The fuzzy prototypes are learned separately on each class by using an unsupervised clustering algorithm based on the possibilistic C-means (PCM) [9]. By this way the prototypes represent an intrinsic description of the classes [12]. Fuzzy prototypes P_i are defined by their membership function $\beta_i(\vec{X})$ (eq. (1)). This one is an hyperellipsoidal Radial Basis Function (RBF) with a center $\vec{\mu_i}$. Its shape is given by a covariance matrix Q_i using the Mahalanobis distance $d_{Q_i}(\vec{X}, \vec{\mu_i})$ [9]:

$$\beta_i(\vec{X}) \quad = \quad 1/(1 + d_{Q_i}(\vec{X}, \vec{\mu_i})). \tag{1}$$

The conclusions a_{ic} of each rule are computed with the pseudo-inverse method [4]. It gives the optimum values to discriminate the classes.

2.3. Recognition

To recognize an unknown character X, its membership degree to all fuzzy prototypes is computed (eq. (1)) and the *sum-product* inference is used to compute the system outputs which are scores s_c for each class.

$$s_c = \frac{\sum_{i=1}^{N} \beta_i . s_c^i}{\sum_{i=1}^{N} \beta_i}, \tag{2}$$

where β_i is the if-part activation of the rule R_i for $\vec{X}$, N is the number of rules and s_c^i is the prototype score given by the rule i for the class c. The decision is given by choosing the class having the best (maximum) score.

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-nn classifiers. It is why the adaptation process proposed here is mainly inspired by the adaptation mechanism of k-nn classifiers [11, 16] i.e. the LVQ principle. But our FIS use RBF and numeric conclusions. So our approach was also guided by EFCL [7] and by the FIS [10] and the RBF [13] learning.

The writer adaptation is done during the use of the system and must respect the embedding constraints. So the presented approach is iterative, i.e. that it uses just the last example written by the user or a buffer containing the last few examples. Furthermore, we suppose here that the adaptation is supervised: each example is labeled correctly. This labeling is possible by asking the user to check the recognition or by using self-supervised technique as in [11].

In the used FIS, the adaptation can be made in several ways in order to better discriminate the classes. The prototypes used in the *if-parts* can be re-centered, distorted, removed. It is also possible to add new ones to take into account the specificities of the writer. The conclusions in the *then-parts* can also be optimized in order to re-estimate the participation of prototypes in the description of each class.

We focus in this study on how to adapt the premises of the system rules by re-centering and re-shaping the prototypes. As these if-parts updates change the description of the input representation space, the FIS consequents must be updated at the same time. We define an *adaptation cycle* as a sequence *premise adaptation* and then *conclusion adaptation* for one example. The update of the numeric conclusions is done with the classical Gradient Descent method which is simple, requires few resources (memory and computing time) and can be used in an iterative mode.

The following section presents in more details the originalities of the approach used to make the *premises adaptation* by re-centering and re-shaping prototypes.

3.1. Premise adaptation: ADAPT

The ***AD**aptation by **A**djustment of **P**roto**T**ypes* (ADAPT) method allows to modify all the prototypes of the FIS by re-centering and re-shaping them for each new example that is inputted. This is done according to their participation in the recognition process.

3.1.1. Prototype re-centering The principle is inspired by the LVQ algorithm [8]. The simplest supervised version (LVQ1) brings the nearest prototype closer to a correctly classified example and moves it away if the example is misclassified. The center $\vec{\mu_i}$ of the prototype P_i is updated according to the displacement vector $\vec{\Delta\mu_i}$:

$$\vec{\mu_i} \quad \Leftarrow \quad \vec{\mu_i} + \vec{\Delta\mu_i} \tag{3}$$

$$\vec{\Delta\mu_i} = \lambda * \delta * (\vec{X} - \vec{\mu_i}), \quad (4)$$

with δ equals to 1 if X is of the same class as P_i and -1 otherwise. The adaptation parameter λ lies between 0 and 1. It controls the amplitude of the displacement and thus the adaptation rate (see discussion in section 3.2).

This method can not be used directly in our FIS. The first reason is that our prototypes are not crisply labeled as they participate in the recognition of all classes. Further more, all prototypes are taken into account to recognize an entry.

In [7] the membership degree is used for EFCL which is a fuzzy unsupervised version of LVQ. In the same way, we propose to refine the updating formula by changing the δ. The update of the prototype P_i of the rule i must improve the score of each class. In this way, there are three reasons to have a significant displacement $\vec{\Delta\mu_i}$: the class score s_c is different from the one expected; the participation s_c^i of the prototype to the final decision is high; and the activation of the premise is high. Equations (5, 6) gives the prototype updating with the proposed ADAPT method:

$$\vec{\Delta\mu_i} = \lambda * \delta' * (\vec{X} - \vec{\mu_r}) \quad (5)$$

$$\delta' = \beta_i * \left(\sum_{c=1}^{C} (b_c - s_c) * s_c^i \right), \quad (6)$$

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

We can rewrite this formula as a sum of displacements where each one increases the class score of the class of the example and decreases the scores of the other classes. So the final updating is a compromise between the improvements of each class score.

3.1.2. Prototype re-shaping 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_i is given by the associated covariance matrix Q_i. So, re-shaping the prototypes corresponds to the re-evaluation of these matrices. Nevertheless, the Mahalanobis distance uses the inverse matrix Q_i^{-1}, so it is more efficient to update directly the inverse matrix than to re-evaluate the covariance matrix and then to inverse it at each adaptation cycle [7].

An unsupervised iterative formula is given by [13] to estimate the inverse covariance matrix :

$$Q_i^{-1} \Leftarrow \frac{Q_i^{-1}}{1-\alpha} - \frac{\alpha}{1-\alpha} \cdot \frac{(Q_i^{-1}\vec{m}) \cdot (Q_i^{-1}\vec{m})^T}{1 + \alpha(\vec{m}^T Q_i^{-1} \vec{m})}, \quad (7)$$

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

The drawback of this 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 [7] where EFCL uses the activation of the prototype, we propose to replace α in equation (7) by α' which uses δ' (eq. (6)):

$$\alpha' = \alpha\delta'. \quad (8)$$

3.2. On-line adaptation strategies

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

A robust adaptation could be obtained by storing all the examples inputted by the user and then by adapting the system using all of them. But it is impossible here because of the limited memory resources. In order to have some diversity of examples and to increase the adaptation speed, the only last F examples are stored in a data buffer. Each time that a new example is inputted, it is added to the data buffer and the oldest one is removed. An adaptation cycle is run for each example stored in this buffer. So F is an adaptation parameter which has an effect on the computing time.

In order to increase the speed and the robustness of the adaptation, we use a classical mechanism [8] consisting in decreasing the λ value (eq. (5)). A high value allows a fast adaptation but makes it unstable. A low one allows a stable and robust adaptation but slower. So a decreasing λ allows a fast and robust adaptation. Here λ decreases from λ_{max} to λ_{min}. The same technique is used for the α of the deformation parameter (eq. (8)).

4. Experiments

In order to validate our approach, we have compared it with an off-line learning using a set of 8 different writers. In this first study, we focus on the global performances of the system, on the adaptation speed (in term of the number of inputted examples), and on the robustness (in term of stability of the recognition rates). Future works will be done to evaluate the computing time, which depends on the choice of the implementation and on the device on which the classifier is embedded.

4.1. Experimental protocol

The experiments are based on the recognition of the 26 isolated lower case Latin letters, without any constraints for the writer. The initial system is learned using a subset of the Ironoff database [15] which contains around 400 writers. 5287 characters were used for the initial learning. The writer specific databases were written on a PDA by 8 users, all different from the Ironoff database. In this experiment, there was no recognition feedback so the writer can not adapt his style to the recognition system. Each writer has

inputted 40 times each characters i.e. 1040 characters per writer. To estimate the adaptation performances for each writer, we proceed by a 4-fold cross-validation technique (called *4-f c-v*). 3/4 of the writer database (780 letters) is used to adapt the system and 1/4 (260 letters) is used to evaluate the results of this adaptation to this personal handwriting style. This adaptation is carried out 5 times with a different order for the adaptation data, to avoid the bias of the letter input order. To observe the adaptation effect during a longer use, the adaptation databases are used twice. The presented results are the average of these 20 tests ($5\times$*4-f c-v*).

The ADAPT method uses the following parameter values that were found experimentally: λ decreases from 0.05 to 0.005 and α decreases from 0.005 to 0.001. The data buffer has a length of $F = 20$. The characters are described by a set of 21 features similar to those used in RESIFCar [1]. The class description is made by defining just one prototype, so the system has 26 rules.

4.2. Global results

The table 1 shows the recognition rates before and after the adaptation ADAPT with the min/max results of the *4-f c-v*. The ADAPT method has very interesting results as it starts with an average recognition rate of 88% and finishes the adaptation at 98.2%. This represents an error reduction of 85%. Furthermore in all the 20 tests there is an improvement as shown by the min/max column.

The results of the adaptation without the re-shaping (Table 1) shows the power of the prototype displacement and the importance of the re-shaping. Indeed with just re-centering 70% of errors are avoided but the complete ADAPT reduces the remaining errors by 50%.

Table 1. Recognition rates (%) before and after on-line adaptation compared with off-line learning and off-line adaptation.

Writ.	Before	ADAPT	max min	ADAPT no reshap.	off-line learning	off-line adaptation
1	88.1	98.7	100. 97.3	97.1	90.3	98.7
2	89.1	97.7	98.5 95.4	95.4	87.0	97.6
3	86.7	96.3	98.5 94.2	95.3	82.7	96.2
4	90.7	99.3	99.6 98.9	97.6	88.5	99.2
5	88.5	98.8	99.6 97.7	96.9	91.4	99.0
6	90.5	97.5	98.9 95.8	96.3	86.4	98.0
7	85.1	98.8	99.6 98.1	95.4	85.6	98.7
8	85.1	98.4	99.2 97.3	97.7	91.7	98.2
aver.	88.0	98.2	99.2 96.8	96.5	88.0	98.2

The ADAPT results are compared with the performances of writer dedicated FIS learned directly with the same available data, for each of the writers using the same *4-f c-v*. Table 1 reports the results of this *off-line learning*. It reveals that there is a lake of data to learn a complete FIS for a writer-dependent classifier. Indeed the average recognition rate is 88% which is the same as the writer-independent FIS. The on-line adaptation method does not present this drawback because the initial recognizer bring with him general knowledge which keep the FIS more general.

The last column of the Table 1 permits to evaluate the on-line adaptation strategy by comparing it with an optimal adaptation strategy where the data are available at once: an *off-line adaptation* with the ADAPT method. So the writer specific adaptation database is used 40 times with a data buffer containing all the examples and the updates of the FIS are applied at the end of each database pass. Doing so each example is used exactly the same number of times as in the on-line adaptation with a data buffer of 20 examples. The on-line adaptation have results comparable with the *off-line adaptation*, so this approach has the same adaptation strength but with lower memory cost and in a transparent way for the user.

4.3. Result analysis

4.3.1. The recognition rates evolution The figure 1 shows the evolution of the average recognition rates for the writer 1 during the adaptation process and the min/max rates of his 20 *4-f c-v* tests.

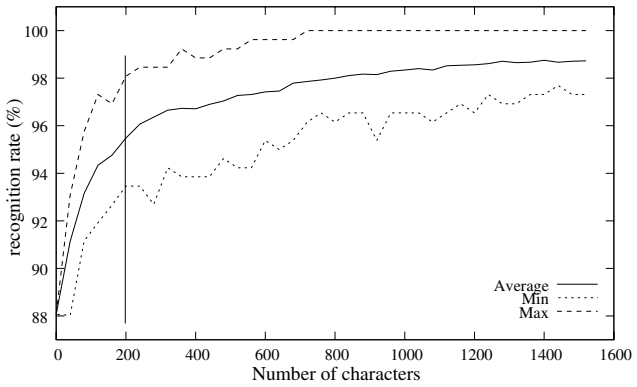
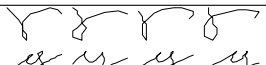
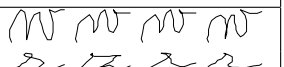
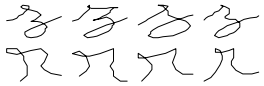
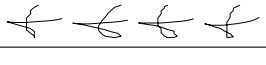
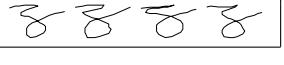


Figure 1. Evolution of the recognition rate during the adaptation.

It shows that the adaptation to the writer's style is very fast: with only 200 characters (about 40 words) the recognition rate rise from 88.1% up to 95.5% and it's about 70% of the final adaptation. Furthermore this adaptation is robust i.e. the recognition rate does not fluctuate during the

Table 2. Some writing styles which are misclassified before adaptation but not after.

Writer		err (%) Bef.	Aft.	Δ	examples		err (%) Bef.	Aft.	Δ	examples
1	'r'	88	1	-99		'w'	55	0	-100	
4	'y'	40	3	-94		'z'	65	6	-92	
7	'q'	93	1	-99		'r'	70	2	-97	
8	'f'	93	6	-94		'z'	68	0	-100	

use. In addition, the difference between the min rate and the max rate tends to be reduced during the adaptation process (from 4.6% after 200 characters to 2.7% at the end).

4.3.2. Style adaptation To observe more specifically the style adaptation, we compare the recognition results of the classifier before and after the adaptation for each writer by showing the writing style examples which are misclassified by the original FIS and correctly classified after the adaptation. The table 2 presents some examples of classes with their error rate before adaptation, their error rate after adaptation, and their relative error rate variation Δ.

Most of these handwriting styles are probably represented in the initial learning database (Ironoff) since it contains numerous writers. But they are confusing with other characters (for example, all errors in the class 'q' of the writer 7 are error confusions with the class 'g'). So the improvement (more than 98% for severals letters) comes from a better representation of the writer style specificities.

5. Conclusion

In the context of fuzzy classifier adaptation for on-line handwritten character recognition, we have presented a new adaptation strategy called ADAPT. This approach is able to adapt prototype based FIS with numeric conclusions.

The reported experiments show good results for 8 different writers. Actually we obtain an error reduction of 85% in average (up to 92.5%) and a recognition rate of 98.2% in average (up to 99.3% for the better). Moreover, examples of writing style illustrate the impact of the adaptation approach to the writer style.

These results are very encouraging. The methods will be extended in future works to the addition and the inhibition of some prototypes and rules. Furthermore, the next step will be to apply these adaptation methods to more complex systems of handwriting recognition such as RESIFCar and Mélidis [3, 12] and to embed them on PDA.

References

[1] E. Anquetil and H. Bouchereau. Integration of an on-line handwriting recognition system in a smart phone device. In *16th ICPR*, volume 3, pages 192–195, 2002.

[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. In *6th IPMU*, pages 259–264, 1996.

[3] E. Anquetil and G. Lorette. On-line handwriting character recognition system based on hierarchical qualitative fuzzy modeling. In *5th IWFHR*, pages 47–52, 1996.

[4] C. Bishop. *Neural Network for Pattern Recognition*. Oxford University Press, 1995.

[5] A. Brakensiek, A. Kosmala, and G. Rigoll. Comparing adaptation techniques for on-line handwriting recognition. In *6th ICDAR*, pages 486–490, 2001.

[6] S. Connell and A. Jain. Writer adaptation for online handwriting recognition. *PAMI*, 24(3):329–346, 2002.

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

[8] T. Kohonen. The self-organizing map. *Proceedings of IEEE*, 78(9):1464–1480, 1990.

[9] R. Krishnapuram and J. Keller. A possibilistic approach to clustering. *IEEE Trans. on Fuzzy Systems*, 1(2):98–110, 1993.

[10] B. Mitaim, S. Kosko. The shape of fuzzy sets in adaptive function approximation. *IEEE Trans. on Fuzzy Systems*, 9(4):637–656, Aug 2001.

[11] L. Oudot, L. Prevost, A. Moises, and M. Milgram. Self-supervised writer adaptation using perceptive concepts : Application to on-line text recognition. In *17th ICPR*, volume 2, pages 598–601, 2004.

[12] N. Ragot and E. Anquetil. Melidis: Pattern recognition by intrinsic/discriminant dual modeling based on a hierarchical organization of fuzzy inference systems. In *10th IPMU*, pages 2069–2076, Perugia, Italy, 2004.

[13] J. Schürmann. *Pattern Classification: a unified view of statistical and neural approaches*. Wiley-Interscience, 1996.

[14] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE TSMC*, 15(1):116–132, 1985.

[15] C. Viard-Gaudin, P. Lallican, S. Knerr, and P. Binter. The irest on/off dual handwrittinf database. In *5th ICDAR*, pages 455–458, 1999.

[16] V. Vuori, J. Laaksonen, and E. Oja. On-line adaptation in recognition of handwritten alphanumeric characters. In *5th ICDAR*, pages 792–795, 1999.