

Fast Incremental Learning Strategy Driven by Confusion Reject for Online Handwriting Recognition

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

Abstract

In this paper, we present a new incremental learning strategy for handwritten character recognition systems. This learning strategy enables the recognition system to learn “rapidly” any new character from very few examples. The presented strategy is driven by a confusion detection mechanism in order to control the learning process. Artificial characters generation techniques are used to overcome the problem of lack of learning data when introducing a new character from unseen class. The results show that a good recognition rate (about 90%) is achieved after only 5 learning examples. Moreover, the rate quickly rises to 94% after 10 examples, and approximately 97% after 30 examples. A reduction of error of 40% is obtained by using the artificial characters generation techniques.

1 Introduction

In the last decade, the world witnessed the emergence of touch screen devices, from Personal Digital Assistants (PDAs) to Tablet PCs and other instruments that use pen-based interfaces. More and more efforts are needed to make online handwriting recognition systems more robust and adaptive in order to meet the increasing user requirements. One of these new requirements is to build a recognition system that allows the user to choose his own group of gestures and assign them to different interactive commands, e.g. “copy”, “paste”, “undo”, etc. This application context imposes specific conditions on the used learning approach. In such approach, the system must be able to rapidly learn a new unseen form using only few data. The later is due to the fact that users would rarely be willing to wait and repeat each new gesture a dozen of times for training the system. Furthermore, such a learning process must be progressively repeated for each newly added data. For all these reasons, this study aims at introducing a fast incremental learning strategy for online handwriting recognition systems. In this

study, the strategy was validated on handwritten characters. However, the long term objective is to use this strategy in any fast incremental gesture learning system.

The main difficulty is to build on-the-fly a handwriting classifier from scratch, with the constraint of few available learning examples, and then to adapt it incrementally in order to achieve high recognition rate as soon as possible.

Incremental learning algorithm is defined in [6] as being one that meets the following criteria: it should be able to learn additional information from new data; it should not require access to the original data (i.e. data used to train the existing classifier); it should preserve previously acquired knowledge (it should not suffer from *catastrophic forgetting*, significant loss of original learned knowledge); and it should be able to accommodate new classes that may be introduced with new data.

Several incremental learning techniques have been presented in different application contexts and for different classification approaches. In the incremental learning approach presented in [6], small batches over separated periods of time are used to add the new knowledge, so it can be considered as a slow and offline incremental learning process. Furthermore, the weak classifiers used in this approach can not be adapted after that they had been learned. Another incremental learning system was presented in [2]. The learning algorithm in this system fits the incremental learning setting described above, and it can be considered as fast learning strategy. However, it uses classifier with basic structure that can not cope with the complexity of handwritten character classification problem.

Indeed, our intention is to design a new strategy that meets both the instantly incremental learning mechanism and the ability of using a complex and powerful classifier structure. The first challenge in designing a prototype-based incremental learning system is to answer these two questions: when and how creating a new prototype?

We presented in previous work [1] an incremental learn-

ing strategy with two distinct phases. A new prototype is created for each new example in the first phase, and no prototype creation is occurred during the second phase. In this paper, we present a homogeneous incremental learning strategy with two fundamental points. The first one is that the prototype creations are completely controlled, during the entire learning process, by the detection of confusion rejection. And the second point is using the original handwritten characters generation techniques, presented in previous work [5], in order to create representative prototypes despite the lack of learning example.

The rest of this article is organized as follows. In Section 2, we present the used classification approach (fuzzy inference system) and the basic principles of our incremental learning system. Then, we describe in Section 3 our incremental learning strategy. Section 4 then describes the method we used to generate artificial characters to accelerate the learning process. Finally, Section 5 shows the results of our experiments.

2 Principles of our approach

2.1 Fuzzy Inference System

We have chosen a Fuzzy Inference System (FIS) as a classification approach for several reasons: first of all, it is a light system, with few of parameters. And also, its flexible nature allows to adapt it easily to absorb a new knowledge.

The classifier is formalized by an order 0 Takagi-Sugeno FIS [7]. The fuzzy rules make a link between intrinsic models that describe the properties of the handwritten characters and the corresponding label. 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 .

The fuzzy prototypes are learned separately on each class by using an unsupervised clustering algorithm. Fuzzy prototypes P_i are defined by their membership function $\beta_i(\vec{X})$ (eq. (1)). This one is an hyper-ellipsoidal Radial Basis Function (RBF) with a center $\vec{\mu}_i$. Its shape is given by a covariance matrix Q_i using the Mahalanobis distance $d_{Q_i}(\vec{X}, \vec{\mu}_i)$:

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

The conclusions a_{ic} of each rule are computed with the pseudo-inverse method or back-propagation algorithm. It gives the optimum values to discriminate the classes.

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.

$$s_c = \frac{\sum_{i=1}^N \beta_i \cdot s_c^i}{\sum_{i=1}^N \beta_i}, \quad (2)$$

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.

2.2 Incremental Learning

There are two manners by which a prototype-based recognition system can be incrementally learned. The first manner is by creating new prototypes for newly available data. The second is by modifying the existing prototypes according to new data.

Although prototype creation helps to improve significantly the classifier capacity (recognition rate), the complexity of the classifier (computing time and memory space) becomes unacceptable if prototypes number becomes sizable. For this reason, the use of prototypes adjustment is essential in order to get a dynamic and light system. Prototypes adjustment (or adaptation) can improve the capacity with same complexity of the classifier, but this technique has some limitations. It is difficult for it to change rapidly the decision borders in the case where the new knowledge (presented by the new data) is little represented by the existing prototypes. In such case, the decision of creating a new prototype is intuitively more reasonable.

We note that both techniques are complementary, and the first challenge to have an efficient incremental learning model is to find the best strategy to combine these two techniques. We suppose that learning and adapting processes are supervised: each example is correctly labeled. This labeling is possible by asking the user to check the recognition or by using self-supervised technique. We present below the techniques of prototype creation and adjustment used in our system.

2.2.1 Prototype creation

Adding a new prototype in a FIS is achieved by adding a new rule. This process is composed of two parts: the creation of the prototype and the calculation of associated conclusions. A prototype is created in the feature space by choosing a center that determines the position of the prototype, and a covariance matrix that represents the size and shape of the prototype.

In non-incremental learning systems, where the system is built using learning dataset with sufficient number of examples, non-supervised classification methods are used to

calculate the prototypes (centers and covariance matrices) for each class. As mentioned in the introduction, one of the fundamental criteria in an incremental learning system is to eliminate the need to use any prior learning data. For this reason, when adding a new rule, a hyper-spherical prototype is creating in the feature space using the only one current example. A first simple solution is to initialize the prototype's center by the current example, and the covariance matrix by a matrix proportional to identity matrix (thus setting the initial size of the prototype). These prototypes are then distorted to have hyper-ellipsoidal shape thanks to the adjustment technique briefly presented below.

2.2.2 Prototypes adaptation

We use in our system the ADAPT method [4]. The Adaptation by Adjustment of ProtoTypes (ADAPT) method allows to modify all the prototypes of FIS by re-centering and re-shaping them for each new example. This is done according to their participation in the recognition process. The principle of ADAPT is inspired by the LVQ algorithm. The method is fully detailed in [4].

3 Our incremental learning strategy driven by confusion reject

We presented in previous work [1] an incremental learning strategy with two distinct phases. During the first phase, that we called fast incremental learning phase, a significant number of prototypes are added in order to rapidly enhance the classifier's performance. Then in the second phase, called adaptation phase, prototype creation is minimized and a classifier adaptation technique (ADAPT) is then used to continue the learning process.

The new strategy that we present in this paper is based on the detection of a confusion reject in order to trigger the new prototype creation process. We aim to limit prototypes number in the system by avoiding creating prototypes in places where there is no likelihood of confusion. This eliminates adding "unnecessary" prototypes for a specific class where this class was already dominant.

The advantage of this strategy is that for "easy to learn" classes, prototype creation will be early limited by the absence of confusions with other classes. However, the system can continue to add new prototypes in order to learn "difficult to learn" classes as long as confusions are triggered. Before presenting the used algorithm, we describe the notion of confusion reject on which this strategy is based.

3.1 Confusion reject

The purpose of the confusion rejection is to assess the reliability of the classifier by detecting patterns for which the classifier is likely to misclassify. These errors are near the

decision boundaries because scores of at least two classes are nearly equal.

Confusion reject can be realized by defining a reject zone on each side of decision boundaries. Each pattern within one of these zones is considered as potential error and is therefore rejected. Figure 1 shows an example in two dimensions of confusion reject zones for three classes. To

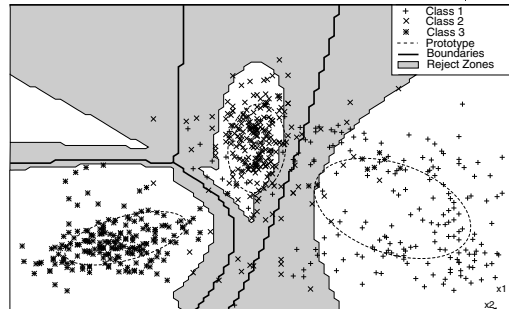


Figure 1. Confusion reject zones.

formalize the confusion reject we use the notion of reliability functions as defined in [3]. Here, the reliability function $\psi(X)$ represents the degree of confusion in classifying a sample X :

$$\psi(X) = (S_{c_1}(X) - S_{c_2}(X))/S_{c_1}(X) \quad (3)$$

where $S_{c_1}(X)$ is the score obtained for the best class and $S_{c_2}(X)$ is the score obtained for the second best class. A sample X is then rejected when the degree of confusion is below a specific threshold. The threshold value represents the width of the reject zones around decision boundaries.

3.2 Algorithm

The principle of this strategy is based on the idea of favoring the creation of prototypes at the beginning, and then progressively reducing the number of creations relying on the adaptation technique to keep up the learning. The Adjustment of the number of created prototypes is provided by changing the threshold value of confusion reject. We use a less strict threshold for a given class each time we add a new prototype for this class. Algorithm 1 describes our incremental learning strategy.

4 Incremental learning acceleration using artificial data generation

Due to lack of knowledge in the beginning of learning a new class, generating synthetic handwritten characters can improve the quality of created prototypes to be more representative of user writing style. Thus, a new prototype is created by calculating the center and the covariance matrix

Algorithm 1: Incremental learning algorithm driven by confusion reject.

```

foreach new example  $e$  do
  if  $e$  is the first example of the class  $C$  then
    create a new prototype around  $e$ ;
    apply adaptation technique;
    reject_threshold[ $C$ ] = initial value;
  else
    apply adaptation technique;
    calculate the confusion degree;
    if confusion reject then
      create a new prototype around  $e$ ;
      reduce the value of reject_threshold[ $C$ ];
    end
  end
end

```

from synthetic characters, which results in hyper-ellipsoidal prototypes instead of hyper-spherical prototypes (created using the original character only).

As presented in [5], several online handwriting generation techniques can be used to build artificial learning dataset, in respecting the user writing style. We use in our system two techniques of characters generation. The first one uses classical image distortions, such as scaling and slanting. The second one is based on the particularity of on-line handwriting; it applies two online distortions on the character: speed variation and curvature modification. Thus, several variations of the same class are generated from a single example of this class. Distortions limits are carefully chosen in order to avoid generating an artificial character that does not look like any more the original one, so it will not help to learn the user writing style.

5 Results

The experiments are based on the recognition of the 26 isolated lower case Latin letter. The writer specific datasets were written on a PDA by 18 writers. Each writer has randomly inputted 40 times each character, i.e 1040 characters per writer. In order to estimate the performance of the incremental learning strategy for each writer, we proceed by a 4-fold cross-validation technique. Three quarters of the dataset (780 letters) are used to incrementally learn the system, and one quarter (260 letters) is used to estimate the evolving of system capacity during the learning process. The presented results in the figures are the average of the 18 tests (18 writers). Each pattern in our system is described by a set of 21 features. A new example of each class is presented to the system in each learning cycle.

We call “strategy 1” the two phases incremental learn-

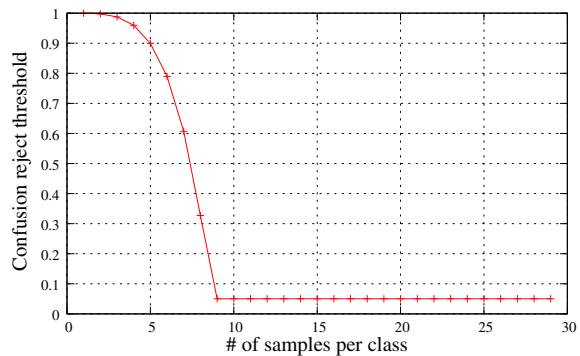


Figure 2. Policy of reducing the confusion reject threshold.

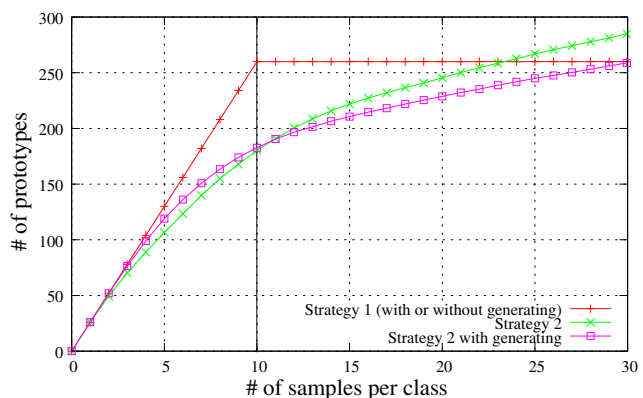


Figure 3. Evolution of total prototypes number during the learning process.

ing strategy that we presented in previous work [1], and “strategy 2” the new incremental learning strategy driven by confusion reject. In the strategy 1, the fast incremental learning phase for each class ends after introducing 10 examples of that class. The system is then switched to the adaptation phase. For the strategy 2, the policy of reducing the confusion reject threshold during the learning process is illustrated in figure 2.

We compare in these experiments the performance of the two incremental learning strategies in terms of the complexity of the classifier, and the quality of the classifier. The complexity of the classifier is related to the required memory space and the computing time required for recognizing one character. In our system, memory space and computing time depend totally on the number of prototypes created in the system. The quality of the classifier is evaluated by the recognition rate obtained with the test datasets.

Moreover, we evaluate in these experiments the impact of using the artificial characters generation on the quality and the complexity of the classifier in our incremental learn-

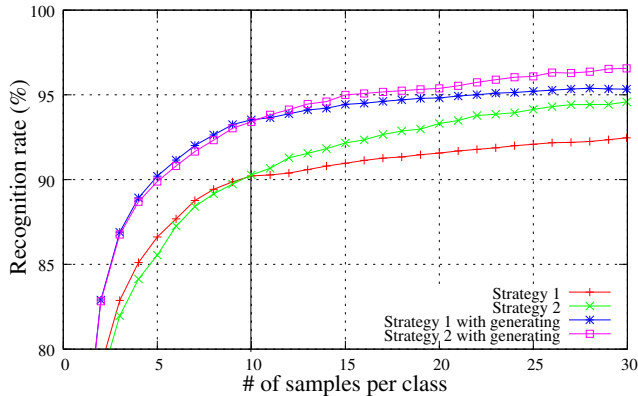


Figure 4. Evolution of recognition rate during the learning process.

ing system. Figure 3 shows the evolution of total prototypes number in the FIS for the 26 classes during the learning process. While figure 4 shows the evolution of the mean recognition rate for the 18 writers.

Through these two figures, we note that the strategy 2 results in a classifier’s quality equal to or greater than that obtained with the strategy 1, with creating fewer prototypes. We find that using the strategy 2 with artificial characters generation, a recognition rate about 90% is reached after only 5 learning examples, and such rate rapidly improves reaching 94% after 10 examples, and about 97% after 30 examples. We note also that an error reduction about 40% is achieved using the artificial characters generation.

Figure 3 shows the impact of the characters generation on the classifier’s complexity. Since prototype creation in strategy 1 is systematic during the phase 1 and inhibited in the phase 2, characters generation has no impact on the number of created prototypes. In contrast, in the strategy 2, the more the system is efficient the less prototype creations are needed. This is confirmed in figure 3 where we note that the use of characters generation with strategy 2 helps to reduce the total prototypes number in the system.

With disregarding the axis of time represented by the increased number of learning examples in figures 3 and 4, we illustrate the quality/complexity ratio of the classifier obtained by each strategy (figure 5). We conclude that strategy 2 achieves a better recognition rate for the same number of prototypes comparing to strategy 1. It can be also noted that the quality/complexity ratio of the classifier is enhanced thanks to the artificial characters generation.

6 Conclusions

In the context of online handwritten character recognition systems, we introduce in this paper a new fast incre-

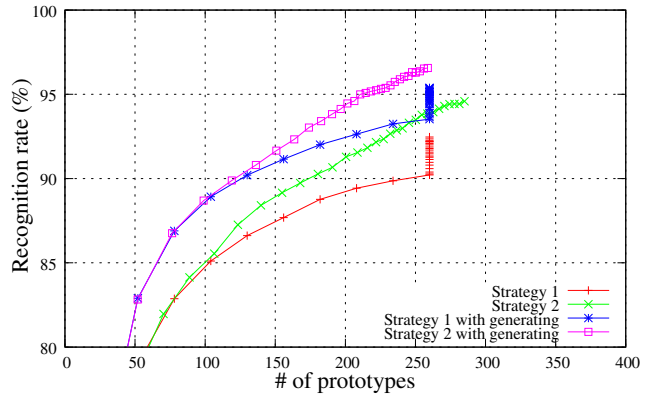


Figure 5. Classifier quality/complexity ratio obtained by each strategy.

mental learning strategy driven by confusion reject mechanism. Using this strategy, the system - which is based on fuzzy inference classifier - is able to learn from scratch new forms with few learning data. It also improves and adapts to each newly available data. Artificial characters generation has been used in the system in order to accelerate the learning by improving the recognition rate. Results have shown that a good recognition rate is achieved after introducing more than 5 examples per class. A special emphasis for a possible future work is placed on reducing the number of prototypes in the system either by deleting the “useless” prototypes or by merging redundant ones.

References

- [1] A. Almaksour, H. Mouchere, and E. Anquetil. Fast online incremental learning with few examples for online handwritten character recognition. In *11th ICFHR*, 2008.
- [2] P. M. Gary G. Yen. An effective neuro-fuzzy paradigm for machinery condition health monitoring. *IEEE Transactions on Systems, Man, and Cybernetics*, 31-4:523 – 536, 2001.
- [3] H. Mouchere and E. Anquetil. A unified strategy to deal with different natures of reject. In *18th ICPR*, 2006.
- [4] H. Mouchere, E. Anquetil, and N. Ragot. Writer style adaptation in on-line handwriting recognizers by a fuzzy mechanism approach : The adapt method. *IJPRAI*, 21(1):99–116, 2007.
- [5] H. Mouchere, S. Bayoudh, E. Anquetil, and L. Miclet. Synthetic on-line handwriting generation by distortions and analogy. In *13th IGS*, 2007.
- [6] R. Polikar, L. Udpa, S. Udpa, and V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 31:497–508, 2001.
- [7] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE TSMC*, 15(1):116–132, 1985.