Fast Online Incremental Learning with Few Examples For Online Handwritten Character Recognition.

Abdullah Almousa Almaksour, Harold Mouchère, Eric Anquetil

Fast Online Incremental Learning with Few Examples For Online Handwritten Character Recognition

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

Abstract

An incremental learning strategy for handwritten character recognition is proposed in this paper. The strategy is online and fast, in the sense that any new character class can be instantly learned by the system. The proposed strategy aims at overcoming the problem of lack of training data when introducing a new character class. Synthetic handwritten characters generation is used for this purpose. Our approach uses a Fuzzy Inference System (FIS) as a classifier. Results have shown that a good recognition rate (about 90%) can be achieved using only 3 training examples. And such rate rapidly improves reaching 96% for 10 examples, and 97% for 30 ones.

Keywords: Incremental learning; online handwritten character recognition; fuzzy inference system; supervised adaptation; handwritten character generation.

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. Online handwriting recognition systems specialists have to keep up with the advancement of such innovations. More and more efforts are needed to make these systems more robust and adaptive in order to meet the rapidly increasing user requirements. One of these new requirements is to build a recognition system that allows the user to choose his/her 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. Such approach must be capable to be rapidly learned 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 in training the system. Further more, such a learning process must be progressively repeated for each newly added data. For all these reasons, this study aims at proposing an original fast incremental learning strategy for online handwriting recognition systems. In this study, the strategy was validated on alphabetical handwritten characters. However, the long term objective is to use such this technique in any fast incremental gesture learning system.

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

In this paper, we propose an incremental learning process with two distinct phases. In order to overcome the problem of lack of examples, and to rapidly increase the system recognition rate, we integrate the original online synthetic handwritten characters generation techniques - presented in previous works [5] - in the learning process.

Generally, for each new learning data in an incremental learning approach, either a new knowledge must be added to the system, or the already existing knowledge has to be adapted to represent this data. In our approach, prototype-based classifier is used for knowledge representation. In the proposed strategy and during the first phase that we call *fast incremental learning phase*, a significant number of prototypes must be added in order to rapidly maximize the capacity of the classifier, using the few available data. Then in the second phase, called *adaptation phase*, prototypes creation is minimized, and a classifiers adaptation technique is then used to enhance the recognition rate.

The rest of this paper is organized as follows. In section 2, we provide an overview of several approaches suggested for incremental learning algorithms. We present our approach in section 3. Next, section 4 describes the fast incremental learning phase; and section 5 describes the adaptation phase. Then, our experimental protocol and results are presented in section 6.

2. Related works

In many practical applications of machine learning systems, acquisition of representative set of training data is considered to be expensive and time consuming. In our context, in which a new classifier must be created from scratch, the time needed to build representative and complete dataset is relatively long and unacceptable from the user's viewpoint. Therefore, such an application scenario requires the classifier to be rapidly trained from a few examples, and then get incrementally updated by any newly available examples.

Incremental learning algorithm is defined in [10] as being one that meets the following criteria:

1. it should be able to learn additional information from new data,
2. it should not require access to the original data (i.e. data used to train the existing classifier),
3. it should preserve previously acquired knowledge (it should not suffer from *catastrophic forgetting*, significant loss of original learned knowledge),
4. it should be able to accommodate new classes that may be introduced with new data.

These four points, which apply for any general incremental learning problem, perfectly match the characteristics of our special problem, a fast learning algorithm for online handwritten characters system. Several incremental learning techniques have been proposed in different application contexts and for different classification approaches, some of which are described below.

Learn++ [10] is an incremental algorithm based on synergistic performance of an ensemble of weak classifiers. It generates multiple hypotheses using training data sampled according to carefully chosen distributions of training dataset into smaller training sub-sets. In this approach, small batches over separated periods of time are used to integrate the new knowledge, so it can be considered as a slow and offline incremental learning process. Further more, the created classifiers can not be adapted after that they had been learned. Therefore, the only method to be used by the incremental learning process in order to add new knowledge is to add new classifiers.

ILFN [4] is an Incremental Learning Fuzzy Neural network. The ILFN classifier uses gaussian radial basis functions to structure decision boundaries. It employs a hybrid of both a supervised and an unsupervised learning schemes to generate its prototypes. This algorithm fits the incremental learning setting described above, and it can be considered as fast learning strategy. However, it uses a basic prototype structure that can not cope with the complexity of handwritten characters classification problem.

Fuzzy ARTMAP [2] is based on generating new decision clusters in response to new patterns that are sufficiently different from previously seen instances. This sufficiency is controlled by a user-defined vigilance parameter. Fuzzy ARTMAP is a versatile algorithm; however, it is sensitive to selection of the vigilance parameter, to the noise levels in the training data and to the order in which the training data is presented to the algorithm.

Indeed, our intention is to design a new strategy that meets both, the instantly incremental learning mechanism as proposed in ILFN, and the ability of using a complex and powerful classifier structure as in Learn++. And so, we can get a fast incremental learning method with a good classification capacity that can solve complex problem, such as handwritten character recognition problem.

3. Our approach

As mentioned above, there are two manners by which the recognition system can be incrementally learned. The first manner is by creating a new prototype for each new example of the character class, the second one of learning is by modifying the existing prototypes that represent the character class according to the characteristics of the new example. In our proposed strategy, we proposed these two methods as complementary techniques to achieve fast and stable increment learning. On one hand, although that prototypes creation helps to improve significantly the recognition rate, its cost can be unacceptable - in term of time and memory - if we continue to create for each new example. On the other hand, prototypes adaptation can slowly enhance the recognition rate, without charging the system by creating new prototypes. As classification mechanism, we have chosen a Fuzzy Inference System (FIS). FIS is a light prototype-based classifier with few parameters. The flexibility of FIS meets our need for an adaptive classification approach.

Furthermore, we suppose here that learning and adapting processes are 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 [9]. In the rest of this section, we present firstly our learning strategy, and then we explain the principals of the used FIS.

3.1. Fast incremental adaptive learning strategy

In the beginning of the learning process, creating new prototypes is indispensable, because the character classes are not yet sufficiently modeled by the existing prototypes. However, even when creating new prototypes for the new examples, the adaptation technique is still useful to adjust the previous prototypes according to these examples. Because of memory and recognition time constraints, prototypes creation must not be continued infinitely. The adaptation algorithm will be the alternative learning methods. After creating sufficient number of prototypes for a specific class, i.e. after introducing N examples of that class,

the system is switched to the adaptation phase. During this phase, prototypes adaptation process will play the main role when introducing a new example. But at the same time, prototypes creation is allowed when a specific number of miss-classified examples are reached. Algorithm 1 describes our learning strategy.

Algorithm 1 Fast incremental adaptive learning algorithm

```
for each new example e from class C do
  if fast incremental learning phase for class C then
    - Apply prototypes creation algorithm;
    - Apply adaptation algorithm in order to adapt already created prototypes according to e;
  end if
  if adaptation phase for class C then
    - Apply adaptation algorithm;
    if e is miss-classified then
      - Increase number of errors for this class nbEr[C];
      if nbEr[C] ≥ specific threshold then
        - Apply prototypes creation algorithm;
        - nbEr[C] = 0;
      end if
    end if
  end if
end for
```

In order to enhance prototypes creation and adaptation processes, the last M examples (about 1 or 2 per class) are stored in a specific buffer. Then, this buffer is used as small training dataset during conclusions calculation and prototypes adaptations, as described later.

3.2. Fuzzy Inference System Principles

We have choose a fuzzy inference system as a classification approach for several reasons: first of all, it is a light system, with few of parameters. And also, its flexible nature allow to adapt it easily to absorb a new knowledge.

3.2.1. Description

The classifier is formalized by an order 0 Takagi-Sugeno FIS [11]. 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.

3.2.2. Learning

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 $d_{Q_i}(\vec{X}, \vec{\mu_i})$ [7]:

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

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

3.2.3. 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.

4. Fast incremental learning phase

In this phase, the incremental learning process consists of two principal algorithms: prototypes creation and conclusion calculation. As mention above, artificial handwriting generation techniques are used in order to achieve high recognition rate by few learning examples.

4.1. Artificial handwriting generation

In order to achieve fast incremental learning and to overcome the lack of data available at the beginning of learning process, there are two strategies to artificially build a training dataset from few samples of writer:

Generating noised-features data. In this strategy, after computing the n-dimensions features which represents the original character, we add noise to each feature value in order to create new points in the classification space. This basic technique can be applied to generate a sufficient learning dataset for the classification models which require a minimum number of examples to represent the new knowledge, for example MLP and SVM based systems. In our system, there is no pre-condition on the size of data in order to create a new prototype; that is because we can create one simple round prototype around the original character point. So technically, this type of data generation is not useful. On the other hand, this mathematical

approach of data generation may not help to create a representative prototype, and it may not be capable to model user writing style.

Generating synthetic handwritten characters. As proposed in [5], different online handwriting generation techniques can be used to build artificial handwriting dataset, in respecting the user style representing by the original example. This kind of generation help the classifier not only to build a sufficiently large data set, but also to improve the recognition rate faster, and finally to help the recognition system to models the writer style without need of a large number of real examples. In [5], three possible strategies to generate synthetic handwritten characters are proposed. 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. The third one is based on the use of analogical proportion on handwriting.

4.2. Prototypes creation

In the used FIS, several fuzzy prototypes have to be created to represent each class (character). In this phase, we start with an empty set of prototypes. So we need to create as much as possible and as soon as possible of prototypes to cover the classification space. Due to the lack of knowledge in the beginning of learning, generating synthetic handwritten characters is an important and indispensable source of knowledge, in order to enhance prototypes creation, in term of quality and quantity.

One prototype will created for each new example of the character during the fast incremental learning phase, until introducing N examples per class. Then the system switches to the adaptation phase, where the prototypes are moved and deformed to accommodate the knowledge acquired by the new examples.

The fuzzy prototypes are learned by using unsupervised clustering algorithm based on the possibilistic C-means. In the fast incremental learning problem, we need to represent and to benefit from all the available knowledge. The unsupervised clustering algorithm neglects the isolated points considering them as noise. Indeed, in our context, these points can help to model user writing style since they result by using online handwriting generation algorithm that model handwriting deformation. To meet this need, we added an enhancement on the clustering algorithm to get it capable to cover the whole cloud of points by the minimum number of possible prototypes .

Unsupervised Classification with Reject Mechanism The reject mechanism proposed by [8] is used in our system to know how many points are covered and well-represented after creating the prototype. A given point is considered rejected if the activations of all created prototypes in the system by this point are less than the reject threshold. The algorithm is presented as following:

1. *Generate np artificial points (characters).*
2. *Apply C-means over original and artificial points to build the principal hyper-ellipsoidal prototype.*
3. ***For** each point p*
   (a) ***If** p is rejected **then** create a new round prototype around the point.*
4. *Calculate the conclusions for newest prototypes.*

4.3. Conclusions calculation

For each newly created prototype, the conclusions are initialized by 1 for the conclusion that corresponds to the character class, and 0 for all other classes. Then, these conclusions are adjusted by applying a back-propagation algorithm over the examples stored in the buffer.

5. Adaptation phase

The writer adaptation is done during the use of the system, even during the fast incremental learning phase. So the presented approach is iterative, i.e. it uses just the last example written by the user and stored in the buffer.

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

5.1. 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. We note that conclusions adaptation is done using back-propagation algorithm over the examples stored in the buffer.

5.1.1. Prototype re-centering

The principle is inspired by the LVQ algorithm [6]. The simplest supervised version (LVQ1) brings the nearest prototype closer to a correctly classified example and moves it away if the example is misclassified. 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.

We propose an adaptation methods called ADAPT, in which all prototype's centers are updated for each new example, and the update of each center μ_i of the prototype P_i must improve the score of each class. In this way, there are three reasons to have a significant displacement $\Delta\vec{\mu}_i$: 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 (3, 4) gives the prototype updating with the proposed method:

$$\vec{\mu_i} \Leftarrow \vec{\mu_i} + \lambda * \delta' * (\vec{X} - \vec{\mu_r}) \quad (3)$$

$$\delta' = \beta_i * \left(\sum_{c=1}^{C} (b_c - s_c) * s_c^i \right), \quad (4)$$

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

The adaptation parameter λ lies between 0 and 1. It controls the amplitude of the displacement and thus the adaptation rate. In order to increase the speed and the robustness of the adaptation, we use a classical mechanism [6] consisting in decreasing the λ value. 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 α from eq. (5) of the prototype reshaping strategy presented in the next section.

5.1.2. 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 [3].

ADAPT proposes an iterative formula to estimate the inverse covariance matrix :

$$Q_i^{-1} \Leftarrow \frac{Q_i^{-1}}{1-\alpha\delta'} - \frac{\alpha\delta'}{1-\alpha\delta'} \cdot \frac{(Q_i^{-1}\vec{m}) \cdot (Q_i^{-1}\vec{m})^T}{1 + \alpha\delta'(\vec{m}^T Q_i^{-1}\vec{m})}, \quad (5)$$

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

5.2. Prototypes creation in adaptation phase

Based on the reject mechanism used in the unsupervised classification, we can divide the character during adaptation phase into four categories: correctly classed and accepted; correctly classed and rejected; incorrectly classed and accepted; incorrectly classed and rejected.

For each class, the characters that belong to the last category are saved. Therefore, when the number of errors for one class exceeds a specific threshold E, a new prototype is created for that class using the miss-classified characters. The artificial character generation is used for the prototype creation. In order to limit the number of prototype creations, the threshold E is incremented after each creation.

6. Experimentation

6.1. Experimental protocol

The experiments are based on the recognition of the 26 isolated lower case Latin letters, without any constraints for the writer. The writer specific databases were written on a PDA by 11 users. 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 learn and adapt the system and 1/4 (260 letters) is used to evaluate the results of the learning process to this personal handwriting style. The presented results are the average of these 11 tests.

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 $M = 2$. The characters are described by a set of 21 features similar to those used in RESIFCar [1]. The class description is made by defining few hyper-ellipsoidal prototypes, and several round prototypes for the rejected points. The fast incremental learning phase for each class finishes after introducing 10 examples of that class. Then, the adaptation technique with more strict prototype creation method is used. For each character, the handwriting generation strategy is used to generate 300 synthetic characters. Image distortions and on-line distortions are used for the generation.

Four different experiments has been made to show the effect of using synthetic characters generation, the reject mechanism for the unsupervised classification, and the adaptation process on the evolution of the average recognition rates.

6.2. Experimental results

Figure 1 illustrates the average recognition rate evolution for the 11 writers. The effect of using the adaptation process and the reject mechanism on this evolution are presented in this figure. Also, this figure represents a comparison between using synthetic handwritten characters generation with regards to noised-features points generation. Table 1 summarizes the experimental results by comparing the recognition rates at different positions during learning process. The number of created prototypes is also presented in this table (in round brackets).

An error reduction of about 25% can be achieved in using synthetic handwritten characters generation, instead of noised-features points generation. It can be noted that eventhough the role of adaptation process is more impor-

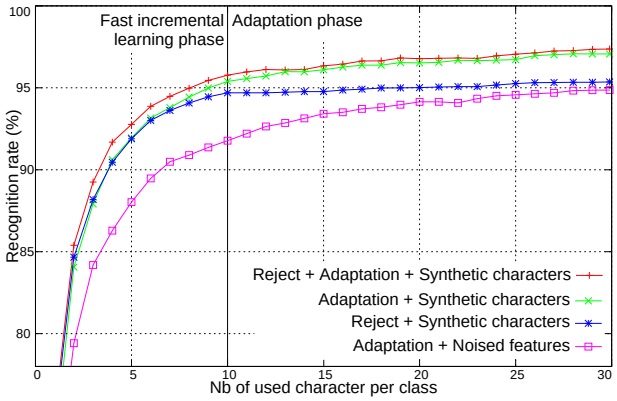

Figure 1. Evolution of the recognition rate during the two learning phases(%)

tant during the adaptation phase, it is also useful during the fast incremental learning phase. For instance, after introducing 3 examples of each class, 10% of recognition errors are avoided by using the adaptation. It can also be mentioned that using the reject mechanism does indeed improve the recognition rate, especially at the early stage of learning. By example, for also 3 characters per class, using the reject reduces the global recognition errors by 11%. We can note that the fact of minimizing prototypes

Table 1. Recognition rates (%) and prototypes count

Nb. characters per class	1	3	10	30
Adaptation & Noised features	70.0 (26)	84.1 (52)	91.7 (260)	94.8 (266)
Adaptation & Synth. characters	73.1 (26)	87.9 (52)	95.3 (260)	97.1 (262)
Reject & Synth. characters	74.6 (198)	88.1 (580)	94.6 (1834)	95.3 (1914)
Reject & Adaptation & Synth. characters	75.1 (200)	89.2 (584)	95.7 (1850)	97.3 (1872)

creation during the adaptation phase has no negative effect on the recognition evolution rate, providing the moment of changing the phase is well chosen.

7. Conclusion and future work

In the context of online handwritten character recognition, we have presented a new strategy for fast online incremental learning based on fuzzy classifier. This approach is able to learn a new character class with few learning data. Results have shown that high recognition rate can be achieved after introducing more than 5 examples per class.

Future works can focus on two points. The first one is the inhibition of some prototypes over the time. At the early stage of learning, a significant number of prototypes are created to rapidly maximize the recognition rate. Some of these prototypes can be later inhibited to enhance system performance. The second point is to use the proposed approach to build a dynamic online handwritten gestures recognition system for real context application.

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] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, and D. Rosen. Fuzzy artmap: A neural network architecture for incrementalsupervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3, 1992.

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

[4] 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.

[5] M. Harold, B. Sabri, A. Eric, and L. Miclet. Synthetic on-line handwriting generation by distortions and analogy. In *13th IGS*, 2007.

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

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

[8] H. Mouchere and E. Anqueti. A unified strategy to deal with different natures of reject. In *18th ICPR*, 2006.

[9] 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.

[10] 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.

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