Combining model-based and discriminative classifiers : application to handwritten character recognition

L. Prevost C. Michel-Sendis A. Moises L. Oudot M. Milgram
Université Pierre & Marie Curie
Laboratoire des Instruments & Systèmes d'Ile de France
Groupe Perception, Automatique & Réseaux Connexionnistes
4 Pace Jussieu, Paris Cedex 75252, Case 164
lionel.prevost@lis.jussieu.fr

Abstract

Handwriting recognition is such a complex classification problem that it is quite usual now to make co-operate several classification methods at the pre-processing stage or at the classification stage. In this paper, we present an original two stages recognizer. The first stage is a model-based classifier that stores an exhaustive set of character models. The second stage is a discriminative classifier that separates the most ambiguous pairs of classes. This hybrid architecture is based on the idea that the correct class almost systematically belongs to the two more relevant classes found by the first classifier. Experiments on Unipen database show a 30% improvement on a 62 classes recognition problem.

1. Introduction

Since 1997, we have worked on dynamic handwriting recognition. We have recently designed a handwriting text recognizer [5] that should be implemented on personal digital assistants or other hand-held devices. For such an application, recognition rates should be very high otherwise it should discourage all the possible users. This can be achieved by making the classifier more accurate.

Focusing on classification errors, there are two situations which reduce the recognition rate.

(a) Unknown dynamics

(b) Ambiguous characters

Figure 1. Ambiguous characters.

- Pattern might be unrelated to the training data. As each user has his own way of writing, many dynamics can appear (figure 1.a). This problem can be overcome by classifying both dynamic and static representations of the character and merging the classification results [7].
- Pattern might be ambiguous (figure 1.b) and some specific pairs of classes constitute the majority of errors made by the classifier like (B,D) or (7,1).

The idea of our hybrid combination method is based on the fact that a given classifier can achieve very good performances in terms of correct recognition rate when considering the two more relevant classes (see table 2). This observation motivates the search for a suitable method that can detect the correct classification among these two classes. This choice results in a two classes (binary) problem.

In section 2, we present the generic methods for character classification and we justify our hybrid approach. Section 3 is devoted to the first stage model-based classifier. Section 4 details the idea of using the second stage discriminative classifier to improve performances. In section 5, we present the concluding remarks and future works.

2. Model generation versus discrimination

There are two standard ways to perform handwriting classification: generate models (and building the so called model-based classifier) or discriminate (figure 2).

Model-based classifiers: during the training stage, one or several models are built independently for each character class. During the test stage, the classification is performed according to similarity between the unknown pattern and the models. These can be neural models [10], markovian models [2] or prototype-based models [1,8,11].

Discriminative classifiers: optimal frontiers between classes are found during the training stage. These are all

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 IEEE

neural classifiers. In Ref. [6], a single network is used to perform discrimination. In Ref. [4] N networks are trained, each one discriminating one class against the $(N-1)$ others. In Ref. [9], $N(N-1)/2$ Discriminative Neural Networks (the so called DNN) are trained to separate pairs of classes (for a N-classes problem). Such a solution seems really promising because it reduces the N-classes problem to two-classes problems. But it is not so relevant when N increases. For example, in the capital letters case, it leads to 325 DNN's creation.

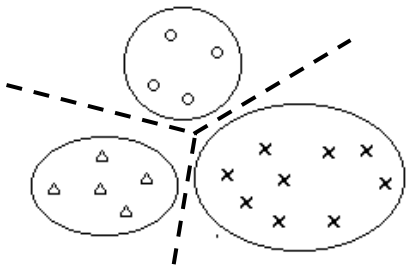


Figure 2. Modelization *vs* discrimination.

Comparing both approaches, it seems relevant to build a hybrid recognizer. A model-based classifier should find the most relevant pair of classes. Then, DNN should improve significantly the accuracy of the model-based classifier in local areas around the frontiers between each pair of ambiguous classes.

3. Model-based classifier

3.1. Prototype-based recognition system

The recognition system used in the experiments is based on prototype matching. It consists of a prototype set covering several writing styles, a dissimilarity measure used for comparing input characters and prototypes, and a decision rule according to which classifications are carried out.

The prototype set was built using MDCA clustering [8]. This algorithm is divided into two stages. The agglomeration stage gathers references around prototypes according to an index of proximity provided by the analysis of inter-reference distance matrix. The adaptation stage optimizes the prototypes and thus improves the character models. Each character model is thus built on a set of prototypes.

The nearest neighbor rule is used as the classification criterion. The distance between the input character and each prototype is computed by dynamic programming. Then for each class, the smallest distance is retained, giving a distance vector $\boldsymbol{D} = (D_1, D_2, \ldots, D_N)$. Assuming that distances are normally distributed, we can compute a posterior probability vector $\boldsymbol{p} = (p_1, p_2, \ldots, p_N)$ and find the most relevant class and the second one.

$C_1 = argmax1(\boldsymbol{p})$ $\quad$ $C_2 = argmax2(\boldsymbol{p})$

The advantage of such a modular architecture (figure 3) over statistical markovian methods and neural networks is based on the fact that it can be adapted quickly: a single character sample is sufficient for learning a new writing style entirely different from those already known.

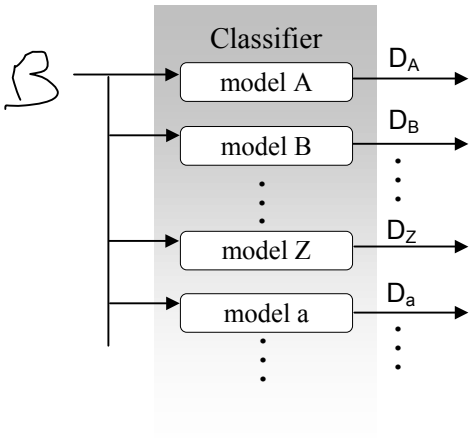


Figure 3. Model-based classifier.

3.2. Database, pre-processing and results

Experiments have been carried out on the Unipen data-set *Train R01-V07* [3], artificially divided in three subsets (table 1), namely the training set S_{TR} (used for clustering), the cross-validation set S_{CV} and the testing set S_{TE}.

Table 1. Data-sets.

	S_{TR}	S_{TE}	S_{CV}
Digit	8000	4000	2000
Uppercase	12826	6355	3188
Lowercase	23922	11443	5974

Characters are simply pre-processed. The sequence of (x,y)-coordinates is resampled with 20 points per stroke, centered and normalized in (-1000,1000) preserving the aspect ratio.

Classification results (table 2) show the robustness of the recognizer and validate our first hypothesis: finding an adequate method to detect confusion among the first and second most relevant classes should increase greatly the system accuracy.

Table 2. Model-based classifier: prototype set and recognition rates (test set S_{TE}).

	S_{proto}	Top 1	Top 2

Digit	476	98.9 %	99.8 %
Uppercase	1158	96.7 %	99.0 %
Lowercase	1114	96.3 %	98.8 %

4. Hybrid classifier

4.1.DNN training process

The model-based classifier generates a probability vector. The first and second higher probabilities indicate the first and second most relevant class (C_1 and C_2). One Discriminative Neural Network can be trained to separate these two classes and detects the most relevant one.

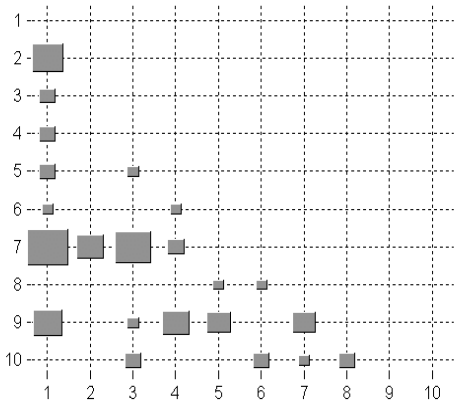


Figure 4. Confusion matrix (Digits: training set S_{TR}).

The pairs of ambiguous classes *(i,j)* can be detected in the confusion matrix. This latter is computed on the digit training set S_{TR} (figure 4) by summing the number of confusions for both pairs *(i,j)* and *(i,j)*. It leads to three observations:

- Many pairs of classes are not ambiguous: training DNN for these pairs of classes is useless;
- Some other pairs are slightly ambiguous: training DNN for these pairs can be useful;
- Some pairs cause the majority of confusions: for example (1,7), (3,7) or (4,9). These pairs need to be discriminated.

So, a confusion threshold δ_{conf} can be used to adjust the number of DNN created and, by the way, the accuracy of the hybrid recognizer.

Given the confusion matrix M_{conf} on the training set S_{TR}, and the number of confusions for each pair $N_{i/j}$, we can compute the total number of confusions N_{tot} and the confusion probability $p_{i/j}$:

$$N_{tot} = \sum_i \sum_j N_{i/j} \qquad p(i,j) = \frac{N_{i/j}}{N_{tot}}$$

We can choose to take into account the most frequent confusions (for which the confusion probability is the highest) i.e confusions for which:

$$p(i,j) > \frac{p_{max}}{\delta_{conf}} \text{ where } p_{max} = max\{p(i,j)\}$$

Table 3 specifies the set of ambiguous pairs *(i,j)* detected for digits for a given threshold δ_{conf}. This set is denoted $S_{\delta conf} = \left\{ (i,j) \middle| p(i,j) > \frac{p_{max}}{\delta_{conf}} \right\}$

For these ambiguous pairs *(i,j)*, a DNN is trained. This is a binary classification problem: *i* will be associated to the DNN label *+1* and *j* will be associated to the other label *-1*.

In table 4, we give the number of DNN created for digits, uppercase and lowercase letters for each confusion threshold. For $\delta_{conf} = 10$, we train 112 DNN. If we try to separate every pairs of classes (even considering separately digits, uppercase and lowercase letters) there should be some 695 DNN.

Table 3. Ambiguous digit pairs *vs* confusion threshold.

S_2	S_4	S_6	S_8	S_{10}
1,2	1,2	1,2	1,2	1,2
1,7	1,7	1,3	1,3	1,3
1,9	1,9	1,4	1,4	1,4
3,7	2,7	1,5	1,5	1,5
	3,7	1,7	1,7	1,7
	4,9	1,9	1,9	1,9
	5,9	2,7	2,7	2,7
	7,9	3,7	3,7	3,7
		3,0	3,0	3,0
		4,7	4,7	4,7
		4,9	4,9	4,9
		5,9	5,9	5,9
		6,0	6,0	6,0
		7,9	7,9	7,9
		8,0	8,0	8,0

Table 4. DNN number *vs* confusion threshold.

δ_{conf}	2	4	6	8	10
Digit	4	8	15	15	15
Uppercase	4	14	18	24	48
Lowercase	9	18	27	39	49
Total	**17**	**40**	**50**	**78**	**112**

4.2. Hybrid classification process

For an unknown (unlabelled) pattern of the test set S_{TE}, two situations can be observed. If the pair (C_1,C_2) does not belong to $S_{\delta conf}$, then we keep the model-based classifier output C_1 . Else, an ambiguity threshold δ_{amb} is

defined and used to activate the DNN. Given the posterior probabilities for the two most relevant classes p_{C1} et p_{C2}, we consider that the model-based classifier output is ambiguous when:

$$\Delta p = p_{C1} - p_{C2} < \delta_{amb}$$

Then, DNN (C_1,C_2) is activated. There are two extrema :

- $\delta_{amb} = 0$: DNN is inhibited and the model-based classifier works alone
- $\delta_{amb} = 1$: DNN is always activated when a confusion (C_1,C_2) is detected.

We show (figure 5) the DNN activation rate that obviously increases when δ_{amb} does.

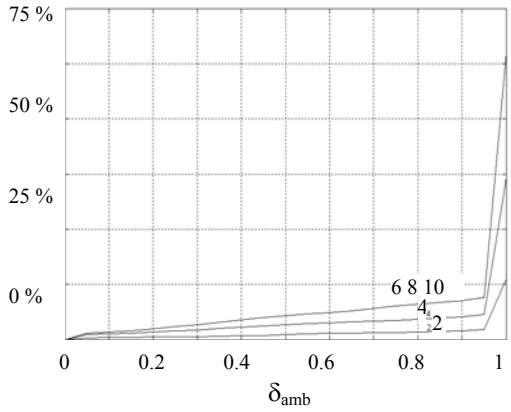


Figure 5. DNN activation rate *vs* ambiguity and confusion thresholds (Digit : test set S_{TE}).

4.3. Database, pre-processings and results

The original training sets for each ambiguous pair (C_i,C_j) do not have the same size and the number of confusions is small. So the frontier between C_i and C_j is not well defined and the corresponding DNN cannot generalize correctly (table 5). Confusions on the training set were detected and transforming them artificially increased their number: each confusion generates new examples slightly expanded, contracted or rotated. So, a modified training set S'_{TR} is generated.

Characters confused by the model-based classifier have always the same number of strokes. In order to get input vectors of the same size, characters are re-sampled to 20 points.

The DNN has the following architecture: 40 cells on the input layer corresponding to the 20 (x,y)-coordinates, one hidden layer and one output cell. The DNN are trained on the modified training sets S'_{TR} and training is stopped on cross-validation set S_{CV}. Several trainings have been done to optimizing the hidden layer size and 10 hidden cells achieve the best recognition rate (table 5).

Table 5. DNN recognition rates on the test set (S_{TE}).

	Original training set	**Modified training set**
Digit	99.7 %	**99.6 %**
Uppercase	99.1 %	**99.3 %**
Lowercase	98.8 %	**99.5 %**

We tested our hybrid recognizer on the test set (S_{TE}: 11162 examples). The following figures 6,7 and 8 summarize the system performances. We can observe that the recognition rate increases when the confusion threshold δ_{conf} does (except for digits, but the number of examples involved is to small to come to a conclusion).

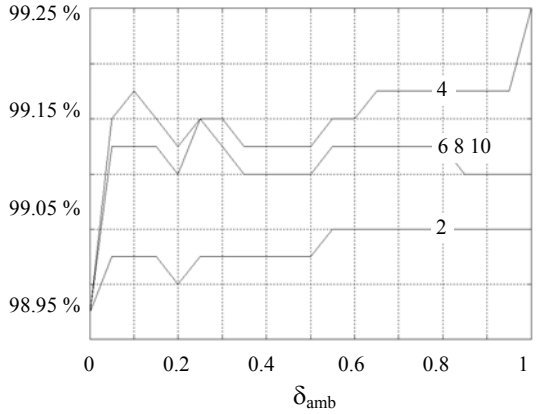


Figure 6. Hybrid classifier recognition rates on digits *vs* ambiguity and confusion thresholds (test set S_{TE}).

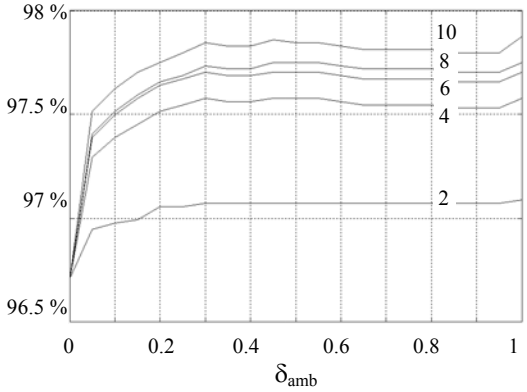
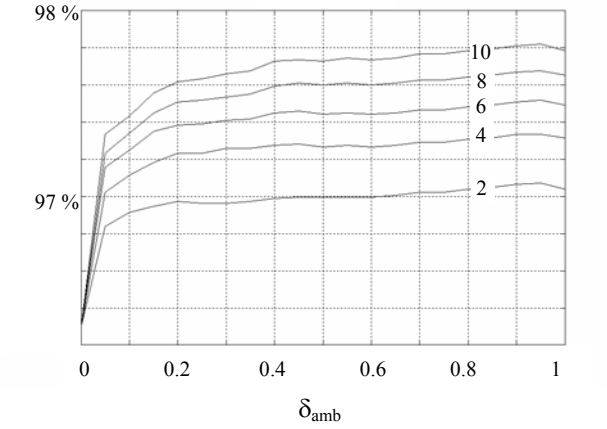


Figure 7. Hybrid classifier recognition rates on uppercase letters *vs* ambiguity and confusion thresholds (test set S_{TE}).

Figure 8. Hybrid classifier recognition rates on lowercase letters *vs* ambiguity and confusion thresholds (test set S_{TE}).

The second observation is that the recognition rate increases in a quasi-monotonous way when the confusion



threshold δ_{amb} does, higher than the model-based classifier rate.

Finally, table 6 summarizes the best performances (with δ_{conf} = 10 and δ_{amb} = 1) of the hybrid classifier compared with the original model-based classifier. These results prove that our hybrid recognizer increases significantly the performances in terms of recognition rate for on-line handwriting classification.

Table 6. Hybrid classifier: Recognition rates (test set S_{TE})..

	Model-based classifier	**Hybrid classifier**
Digit	98.9 %	**99.1 %**
Uppercase	96.7 %	**97.8 %**
Lowercase	96.3 %	**97.8 %**

5. Conclusions

The main idea of this paper is that the recognition process can be performed in 2 steps. First one is a competition between all classes and just leads to "happy fews" (2 for instance) selected classes. This kind of "short list" cannot be just processed by a competition between several models but needs discrimination. The reason is that remaining ambiguity is concentrated in very specific and local features (think to the difference between uppercase B and D). We have clearly shown in this work that this kind of hybridation works and we shall experiment other mixture of specialized experts to improve our classifier accuracy.

6. References

[1] Anquetil E. & Lorette G., On-line Handwriting Recognition system Based on Hierarchical Qualitative Fuzzy Modeling, *IWFHR'96, pp 47-52, 1996.*

[2] Connell S. & A.K. Jain, Writer adaptation of on-line handwriting models, *ICDAR'99, pp 434-437, 1999.*

[3] Guyon I., Schomaker L., Plamondon R., Liberman M. & Janet S., UNIPEN project of on-line data exchange and recognizer benchmarks, *ICPR'94, pp. 29-33, 1994.*

[4] Oh I.S. & Suen C.Y., A class-modular feed-forward neural network for handwriting recognition, *Pattern Recognition, 35, pp 229-244, 2002.*

[5] Oudot L., Prevost L. & Milgram M., Dynamic recognition in the omni-writer frame : application to hand-printed text recognition, *ICDAR'2001, http://icdar.djvuzone.org/jss/index.html, 2001.*

[6] Poisson E., Viard-Gaudin C. & Lallican P.M., Combinaison de réseaux de neurones à convolution pour la reconnaissance de caractères manuscrits en-ligne, *CFD'02, pp 315-324, 2002.*

[7] Prevost L. & Milgram M., Static and dynamic classifier fusion for character recognition, *ICDAR'97, (2), pp499-506, 1997.*

[8] Prevost L. & Milgram M., Modelizing character allographs in omni-scriptor frame : a new non-supervised algorithm, *Pattern Recognition Letters, 21(4), pp 295-302, 2000.*

[9] Price D., Knerr S., Personnaz L. & Dreyfus G., Pairwise neural network classifiers with probabilistic outputs, *NIPS, 7, 1994.*

[10] Schwenk H. & Milgram M., Constraint tangent distance for on-line character recognition, *ICPR'96, (D), pp 520-524, 1996.*

[11] Vuori V., Laaksonen J. & Kangas,J., Influence of erroneous learning samples on adaptation in on-line handwriting recognition, *Pattern Recognition, 35(4), pp 915-925, 2002.*