Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Pattern Recognition Letters xxx (2005) xxx–xxx
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Hybrid generative/discriminative classifier for unconstrained character recognition

Lionel Prevost *, Loïc Oudot, Alvaro Moises, Christian Michel-Sendis, Maurice Milgram

Université Pierre et Marie Curie, Laboratoire des Instruments & Systèmes d'Ile de France, Groupe PARC 4 Pace Jussieu, Paris Cedex 75252, Case 252, France

Abstract

Handwriting recognition for hand-held devices like PDAs requires very accurate and adaptive classifiers. It is such a complex classification problem that it is quite usual now to make co-operate several classification methods. In this paper, we present an original two stages recognizer. The first stage is a model-based classifier which store an exhaustive set of character models. The second stage is a pairwise classifier which separate 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 a 80,000 examples database show a 30% improvement on a 62 classes recognition problem. Moreover, we show experimentally that such an architecture suits perfectly for incremental classification.

Keywords: Handwriting recognition; Multiple classifier system; Pairwise neural networks; Confusion matrix; Adaptive classifier

1. Introduction

Recently, hand-held devices like PDAs, mobiles phones or e-books have became very popular. In opposition to classical personal computers, they are very small, keyboard-less and mouse-less. Therefore, electronic pen is very attractive as pointing and handwriting device. The first application belongs to man-machine interface and the second to handwriting recognition. Here, we focus on the second one.

For such an application, recognition rates should be very high otherwise it should discourage all the possible users. The major problem is the vast variation in personal writing style. This problem can be solved either by constraining the al-

* Corresponding author. Tel.: +33 1 44 27 23 48; fax: +33 1 44 27 62 14.
E-mail address: lionel.prevost@lis.jussieu.fr (L. Prevost).

0167-8655/$ - see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.patrec.2005.03.005

lowed style of writing (PDA's *grafiti* alphabet: Fig. 1a), trying to learn all personal writing styles (natural and script writing: Fig. 1b and c) to build an omni-writer recognizer or building a mono-writer recognizer by adapting the system to its user' style and habits (abbreviations, mathematical or chemical symbols for scientists...).

In dynamic handwriting recognition, signal is represented by sequences of (x, y) coordinates of the pen moving. Each handwriting style has got its typical allographs. This notion, particular to handwriting, includes on the one hand characters having the same image but presenting a very variable dynamics in term of the number of stroke composing the character, their senses and direction and on the other hand, the different handwriting model of a given character: cursive, hand-printed, mixed...

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

- Pattern might be unrelated to the training data. As each user has his own way of writing, many dynamics can appear (Fig. 2a) This problem can be overcome by classifying both dynamic and static representations of the character and combining the classification results as shown in (Prevost and Milgram, 1997).
- Pattern might be ambiguous (Fig. 2b) 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

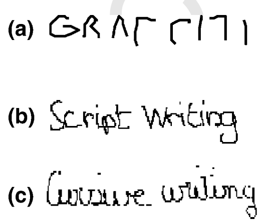


Fig. 1. Handwriting styles. (a) Constraint writing, (b) script writing, (c) natural writing.

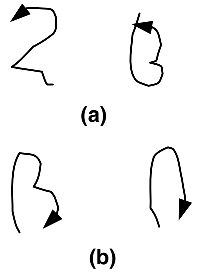


Fig. 2. Ambiguous characters. (a) Unknown dynamics, (b) ambiguous characters.

classes (see Table 2). This observation motivates the search for a suitable method which can detect the correct classification among these two classes. This choice results in a two class (binary) problem. In this paper, we explore the following combination scheme. First stage generative classifier is used to detect ambiguous pairs of classes and the second stage is discriminative. It is composed of a set of pairwise neural networks, one for each ambiguous pair of classes.

The paper is organized as follows. Section 2 describes several standardized methods used for character classification and ways to build two stage classifier well-known for their accuracy. Section 3 details the first stage model-based classifier. Section 4 is devoted to the implementation of the second stage discriminative classifier used to improve performances. In Section 5, we show that this hybrid approach is adaptive. Finally, concluding remarks and future works are discussed in Section 6.

2. Review of classification methods

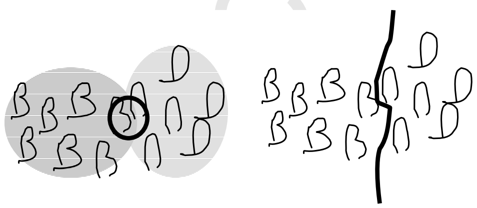
2.1. Model generation vs discrimination

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

Generative classifiers train one (or several) model(s) for each character class with examples of this single class. During the test stage, the classification is performed according to similarity between the unknown pattern and the models. Neural models (Schwenk and Milgram, 1996), Markovian models (Connell and Jain, 1999) or prototype-based models (Anquetil and Lorette, 1996; Prevost and Milgram, 2000; Vuori et al., 2002) are usual to perform handwriting recognition (Plamondon and Srihari, 2000). All these models are perfectly fitted to multi-modal classes and can face the variability of handwriting. Their main drawback is the possibility of model overlapping when classes are ambiguous (Fig. 3a).

Discriminative classifiers find optimal frontiers between classes (Fig. 3b). These classifiers are trained using examples of a given set of classes (two, or more). Guyon et al. (1991) use a single network to perform discrimination. Oh and Suen (2002) apply a "class-modular" (also called "one against all") strategy and train N networks. Each classifier is trained to discriminate one class from all the remaining classes ($N - 1$). They show that this approach is more accurate than the single network approach, especially when the number of class increases. Another method is the "pairwise coupling", which consists to construct a classifier for each pair of classes. Thus, a N class problem is decomposing in $N(N - 1)/2$ binary sub-problems. Price et al. (1994) train pairwise neural networks (the so called PNN) 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 PNN's creation. The question is: is it necessary to train a discriminative classifier when data can be easily discriminated? and, in such a case, should a model-based classifier be sufficient?

Comparing both approaches, it seems relevant to build a hybrid recognizer to obtain the best of both words (Raina et al., 2003). Teo and Shinghal (1997) combine a ruled-based classifier with PNN to confirm or reject data. OCR applications are also presented by Avi-Itzhak et al. (1995) and Wang and Jean (1993) to solve multifont character recognition and confusions with hierarchical NN. In our approach, a model-based classifier find the most relevant pair of classes. Then, PNN are used to improve significantly the accuracy of the model-based classifier in local areas around the frontiers between each pair of ambiguous classes.



(a) (b)

Fig. 3. Generative classifier (a) and discriminative classifier (b).

2.2. Two stage classification

As shown before, the reason why handwriting recognition is so difficult is the huge style variation and variability in handwriting, leading to the conclusion that a single classifier cannot recognize all kinds of handwriting. The combination of different classifiers, trained on distinct features and/or different data has been applied successfully on handwriting recognition as discussed in recent reviews (Jain et al., 2000; Rahman and Fairhust, 2003). Here, we focus on a particular combination scheme: the two stage classifier. In his taxonomy, Rahman describes the multi-stage classifier as a vertical decision combination where "at each stage there is only one classifier operating and processing the patterns". The class set reduction approach is one of the basic strategy to deal with such a combination. At each stage, the classifier generates a list of the set of possible classes the current character might belong. The next classifier limits further investigations to this subset and produces a reduced list and so on.

Bellili et al. (2003) developed an hybrid MLP-SVM classifier. The first stage is a single neural network (MLP). Ambiguous character pairs are found by computing its confusion matrix and estimating the error probability for each pair. The second stage is a set of local, pairwise SVMs (one for each ambiguous pairs) tuned to detect the correct class among the MLP's two maximum output. Mark that both stages are discriminative. Milgram et al. (in press) pre-estimates probabilities in the first stage with a model-based approach an re-estimates only the highest probabilities with appropriate SVM in the second stage. The author try to get the best of both generative and discriminative classifiers. In (Vuurpijl et al., 2003), several ways to detect conflicts in any first stage classifier or ensemble of classifiers are described. Majority vote is explored, to activate pairwise SVM again. In fact, these two stage classifiers can be seen as particular extension of the probabilistic AdaBoost (Freund and Schapire, 1997), where two classifiers are tuned on different training sets. The second stage training set depends on the performances of the first one and the second stage classifier (or ensemble of classifiers) focuses on local areas around the frontiers between 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 (Fig. 4).

The prototype set was built using the MDCA clustering algorithm described in (Prevost and Milgram, 2000). 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. Thus, each character model is composed of a set of prototypes.

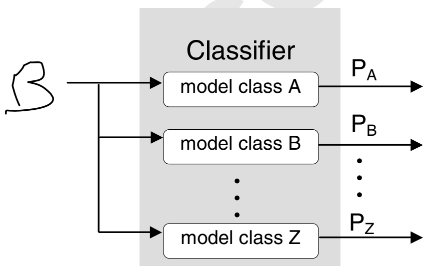


Fig. 4. Model-based classifier.

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 (one distribution per class, parameters: m_i, σ_i estimated on the reference dataset), 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.

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

3.2. Database, pre-processing and results

Experiments have been carried out on the Unipen dataset *Train R01-V07* initiated by Guyon et al. (1994), artificially divided into 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}. Characters are simply pre-processed. The sequence of (x, y)-coordinates is resampled with 20 points per stroke, centered and normalized in $(-1, 1)$ preserving the aspect ratio.

Table 2 gives the size of the prototype set S_{proto} after clustering and the recognition rates on the test set considering the correct class is respectively the first answer (top 1) or one of the two best answers (top 2). It shows the robustness of the recognizer and validates our first assumption: finding an

Table 1
Data-sets

	S_{TR}	S_{TE}	S_{CV}
Digit	8000	4000	2000
Uppercase	12,826	6355	3188
Lowercase	23,922	11,443	5974

Table 2
Model-based classifier: size of the 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

PATREC 3664 | ARTICLE IN PRESS | No. of Pages 9, DTD = 5.0.1
1 April 2005 Disk Used

adequate method to detect confusion among the first and second most relevant classes should increase greatly the system accuracy.

4. Hybrid classifier

4.1. PNN 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 pairwise neural network can be trained to separate these two classes and detects the most relevant one. Under the assumption that, first, the behavior of the classifier is characterized by a confusion matrix and, second, that its prior behavior is representative of its posterior behavior, the pairs of ambiguous classes (i,j) can be easily detected. The confusion matrix is computed on the digit training set S_{TR} (Fig. 5). As we focus on confusions, the diagonal (which indicates correct classifications) is set to zero and the number of confusions $N_{i/j}$ for each pair of classes can be computed by summing the number of confusions for both pairs (i,j) and (j,i).

We can make three observations:

- many pairs of classes are not ambiguous: training PNN for these pairs of classes is useless;
- some other pairs are slightly ambiguous: training PNN 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 PNN created and, by the way, the accuracy and speed 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 $p_{i/j}$:

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

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

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

Table 3 specifies the set of ambiguous pairs (i,j) detected for digits for a given threshold δ_{conf}. This set is denoted

$$S_{\delta_{\text{conf}}} = \left\{ (i,j) \middle| p(i,j) > \frac{p_{\max}}{\delta_{\text{conf}}} \right\}$$

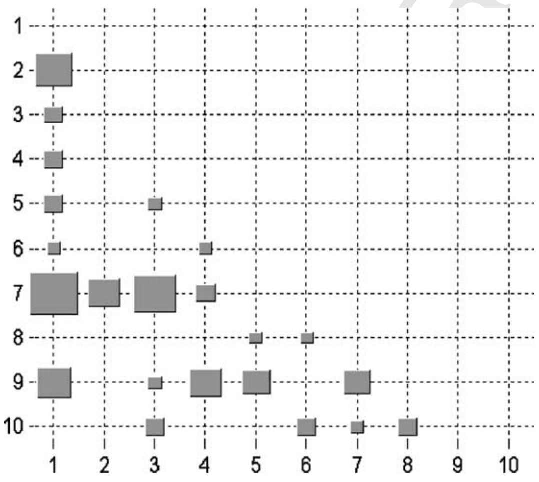


Fig. 5. Confusion matrix (digit: training set S_{TR}).

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

For these ambiguous pairs (i,j), a PNN is trained. This is a binary classification problem: i will be associated to the PNN label +1 and j will be associated to the other label −1. In Table 4, we give the number of PNN created for digits, uppercase and lowercase letters for each confusion threshold. For $\delta_{\text{conf}} = 10$, we train no more than 112 PNN. If we try to separate every pairs of classes (even considering separately digits, uppercase and lowercase letters) there should be some 695 PNN.

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 $S_{\delta_{\text{conf}}}$, then we keep the model-based classifier output C_1. Else, an ambiguity threshold δ_{amb} is defined and used to activate the PNN. Given the posterior probabilities for the two most relevant classes p_{C_1} and p_{C_2}, we consider that the model-based classifier output is ambiguous when: $\Delta p = p_{C_1} - p_{C_2} < \delta_{\text{amb}}$.

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

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

We show (Fig. 6) the PNN activation rate which obviously increases when δ_{amb} does.

4.3. Database, pre-processings and results

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

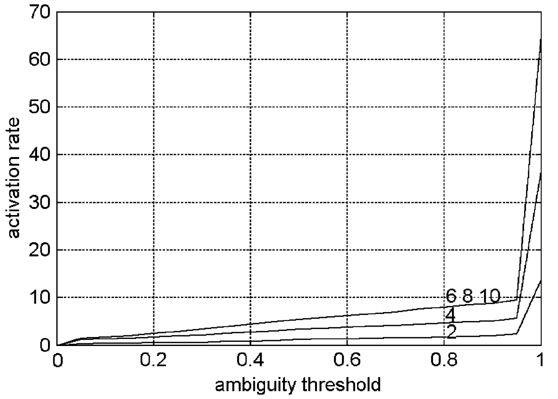


Fig. 6. PNN activation rate vs ambiguity and confusion thresholds.

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.

PNN are completely connected multi-layer perceptron and have the following architecture: 40 cells on the input layer corresponding to the 20 (x, y)-coordinates, one hidden layer and one output cell. PNN are trained using the back-propagation algorithm on the modified training set S'_{TR} and training is stopped on cross-validation set S_{CV}. Several trainings have been done to optimize the hidden layer size and 10 hidden cells achieve the best recognition rate (Table 5). This latter is slightly better when using the modified training set.

Table 4
PNN 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

Table 5
PNN 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}: 21,800 examples). The following Figs. 7–9 summarize the system performances on digits, uppercase and lowercase letters. 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). The second observation is that the recognition rate increases in a quasi-monotonous way when the ambiguity 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.

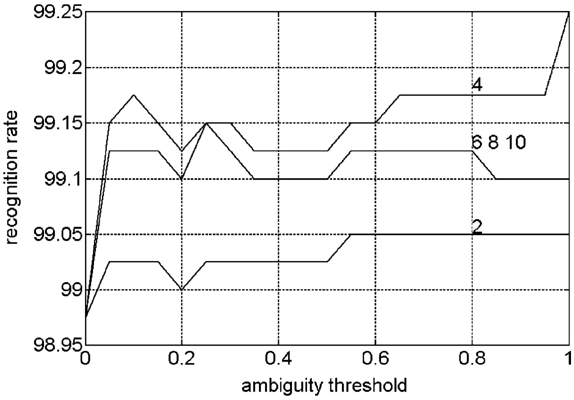


Fig. 7. Hybrid classifier recognition rates on digits vs ambiguity and confusion thresholds (test set S_{TE}).

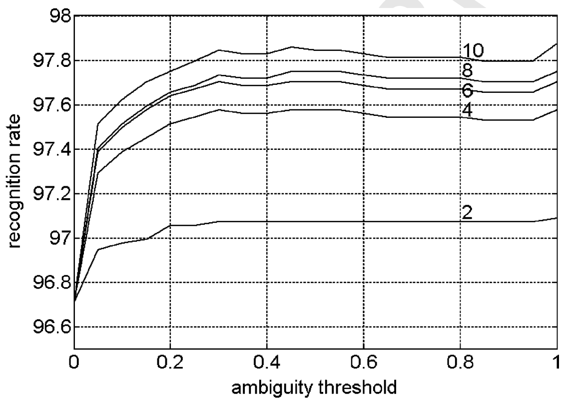


Fig. 8. Hybrid classifier recognition rates on uppercase vs ambiguity and confusion thresholds (test set S_{TE}).

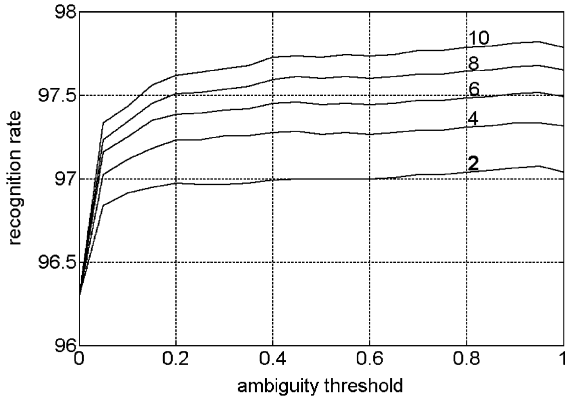


Fig. 9. Hybrid classifier recognition rates on lowercase vs ambiguity and confusion thresholds (test set S_{TE}).

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

	Model-based classifier (%)	Hybrid classifier (%)
Digit	98.9	99.1
Uppercase	96.7	97.9
Lowercase	96.3	97.8

5. Adaptive classifier

In this section, we try to show that our hybrid recognizer is perfectly suited for adaptive learning i.e. can support the addition of new classes. This preliminary study is just a simulation: we consider that the former classifier just recognized 26 uppercase letters and we try to add 10 new classes corresponding to digits.

5.1. Adapting the model-based classifier

Thanks to its structure, the model-based classifier can easily be adapted to including new classes. In opposite to discriminative classifiers (which need a complete re-training), generative classifiers are naturally incremental as model for one class is trained with data of this sole class. Once trained, the new models are included in the classifier and the decision is taken considering the former and the new models (Fig. 10).

5.2. Adapting the hybrid classifier

The adaptation is made as follows:

(1) The extended confusion matrix including all the 36 classes is computed on the training set. As we can see (Fig. 11) cross-confusions between uppercase letters and digits like (O, 0), (I, 1), (S, 5) and (Z, 2) are obviously numerous. In context-free recognition, this kind of confusions cannot be found. So, we choose to delete these latter and zoom-up on cross-confusions (Fig. 12).
(2) Confusions between former classes and new classes (the so-called cross-confusions) are detected, using a confusion threshold $\delta_{conf} = 10$.
(3) Corresponding PNN are trained.

Table 7 shows the recognition rates in the following cases:

- Model-based classifier alone: recognition rate falls dramatically owing to cross-confusions between uppercase letters and digits like (O, 0), (I, 1), (S, 5) and (Z, 2) for example.
- Former hybrid classifier: confusions between uppercase letters (U, U) and digits (D, D) are considered but cross-confusions are ignored.
- Adapted hybrid classifier: all the confusions including cross-confusions (U, D) are taken into account. The result is impressive: the error rate has been reduced by half when compared with the former model-based classifier. PNN have focused on local features to discriminate confusive characters.

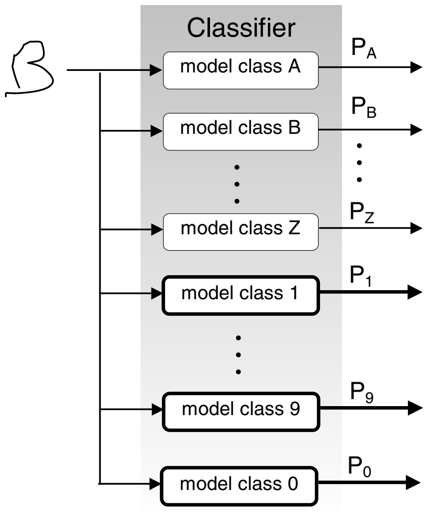


Fig. 10. Adapted model-based classifier.

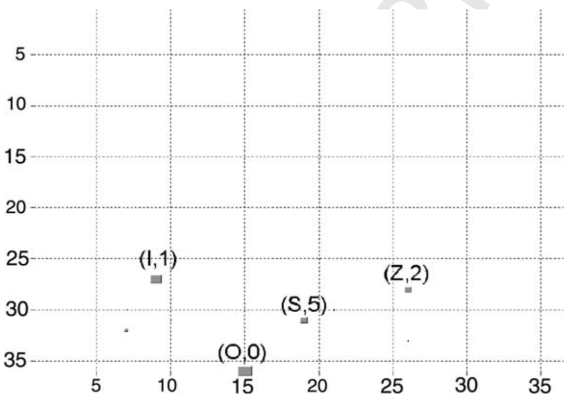


Fig. 11. Extended confusion matrix (uppercase letters and digits: training set S_{TR}).

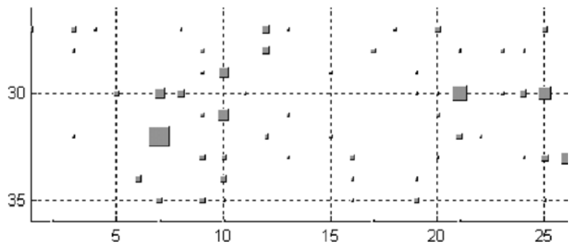


Fig. 12. Cross-confusions (uppercase letters (1–26) and digits (27–36)).

Table 7
Adapted hybrid classifier: recognition rates (test set S_{TE}) and PNN number

	Model-based classifier	Hybrid classifier	
		(U, U) (D, D)	(U, U) (D, D) (U, D)
Reco. rate	90.3%	91.5%	95.4%
PNN	–	63	79

6. Conclusions

The main idea of this paper is that the recognition process can be performed in two stages. The first classifier 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). If data are available, they can be used to determine cases where mistakes are made and train pairwise, local classifiers.

Moreover, the proposed architecture suits perfectly to incremental classification: contrary to discriminative classifiers, this combined generative-discriminative classifier can be adapted to new classes without a complete retraining. It just need to estimate the new models (generation step), detecting cross-confusions and training the corresponding PNN.

We have recently designed a handwriting text recognizer (Oudot et al., 2004) and the proposed recognizer should be integrated into the system. Nevertheless, a question remains: at the present time, the second stage classifier uses a large database to determine pairwise confusions and to train the pairwise neural networks, what will be the training result on a small set of confused examples?

References

Anquetil, E., Lorette, G., 1996. On-line handwriting recognition system based on hierarchical qualitative fuzzy modeling. International Workshop on Frontier of Handwriting Recognition, pp. 47–52.

Avi-Itzhak, H.I., Diep, T.A., Garland, H., 1995. High accuracy optical character recognition using neural networks. IEEE Trans. Pattern Anal. Mach. Intell. 17 (2), 218–224.

Bellili, A., Gilloux, M., Gallinari, P., 2003. An MLP-SVM combination architecture for off-line handwritten digit recognition. Int. J. Document Anal. Recogn. 5, 244–252.

Connell, s., Jain, A.K., 1999. Writer adaptation of on-line handwriting models. International Conference on Document Analysis and Recognition, pp. 434–437.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Computer System Sci. 55, 119–139.

Guyon, I., Albrecht, P., LeCun, Y., Denker, J., Hubbard, W., 1991. Design of a neural network character recognizer for a touch terminal. Pattern Recogn. 24 (2), 105–119.

Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., Janet, S., 1994. UNIPEN project of on-line data exchange and recognizer benchmarks. International Conference on Pattern Recognition, pp. 29–33.

Jain, A.K., Duin, R., Mao, J., 2000. Statistical pattern recognition: A review. IEEE Trans. Pattern Anal. Mach. Intell. 22 (1), 4–37.

Milgram, J., Cheriet, M., Sabourin, R., in press. Speeding up the decision making of support vector classifiers. International Workshop on Frontier of Handwriting Recognition.

Oh, I.S., Suen, C.Y., 2002. A class-modular feed-forward neural network for handwriting recognition. Pattern Recogn. 35, 229–244.

Oudot, L., Prevost, L., Milgram, M., 2004. An activation-verification model for on-line text recognition. International Workshop on Frontier of Handwriting, pp. 9–13.

Plamondon, R., Srihari, S.N., 2000. On-line and off-line handwriting recognition: a comprehensive survey. IEEE Trans. Pattern Anal. Mach. Intell. 22 (1), 63–84.

Prevost, L., Milgram, M., 1997. Static and dynamic classifier fusion for character recognition. International Conference on Document Analysis and Recognition (2), pp. 499–506.

Prevost, L., Milgram, M., 2000. Modelizing character allographs in omni-scriptor frame: A new non-supervised algorithm. Pattern Recogn. Lett. 21 (4), 295–302.

Price, D., Knerr, S., Personnaz, L., Dreyfus, G., 1994. Pairwise neural network classifiers with probabilistic outputs. Neural Inform. Process. Systems 7.

Rahman, A.F., Fairhust, M.C., 2003. Multiple classifier decision combination strategies for character recognition: A review. Int. J. Document Anal. Recogn. 5, 166–194.

Raina, R., Shen, Y., Ng, A.Y., McCallum, A., 2003. Classification with hybrid generative/discriminative models. Neural Inform. Process. System 16.

Schwenk, H., Milgram, M., 1996. Constraint tangent distance for on-line character recognition. International Conference on Pattern Recognition (D), pp. 520–524.

Teo, R.Y.-M., Shinghal, R., 1997. A hybrid classifier for recognizing unconstrained numerals. International Conference on Document Analysis and Recognition, pp. 283–287.

Vuori, V., Laaksonen, J., Kangas, J., 2002. Influence of erroneous learning samples on adaptation in on-line handwriting recognition. Pattern Recogn. 35 (4), 915–925.

Vuurpijl, L., Schomaker, L., Van Erp, M., 2003. Architecture for detecting and solving conflicts: Two stage classification and support vector classifiers. Int. J. Document Anal. Recogn. 5, 213–223.

Wang, J., Jean, J., 1993. Resolving multifont character confusion with neural networks. Pattern Recogn. 26 (1), 175–188.