Cascade Classifier : Design and Application to Digit Recognition

Jean Duong, Hubert Emptoz
Laboratoire d'InfoRmatique en Images et Systèmes d'information (LIRIS)
Institut National des Sciences Appliquées (INSA) de Lyon
20 avenue Albert Einstein, 69621 Villeurbanne CEDEX (France)
jean.duong@liris.cnrs.fr, hubert.emptoz@liris.cnrs.fr

Abstract

The focus of this paper is on machine learning. More specifically, a classifier combination is proposed. We name it "cascade classifier". Conceptually, this system is based on confusion analysis and local re-training mechanism. We describe its architecture. Experiments with handwritten digit images of the MNIST ("Modified NIST") database are presented.

1. Introduction

Many computer applications often need classification tools to process large amounts of data. Advances in pattern recognition field result in different kinds of classifiers to solve various problems *e.g.* neural networks, fuzzy systems, bayesian networks, makovian processes, support vector machines, *etc*. The higher the accuracy values are expected, the more difficult it becomes to make an *a priori* choice between all these possible methods. Even with a given architecture, more or less numerous parameters are to be set for the chosen "machine". This task can become tedious for a non expert user.

In this paper, we propose a classifier combination mainly based on data analysis techniques. Statistics paradigm may be considered as highly reliable since a rigorous mathematical background exists. Moreover, it is widely used in various research fields from biology to economics and many other ones.

Our presentation is organized as following. In section 2, we describe a combination called cascade classifier. Some bibliographical remarks are issued in section 3. Experiments involving a handwritten digit database are proposed in section 4, leading to conclusion in section 5.

2. Cascade classifier

2.1 Data

Let $\mathcal{X}$ be a finite set of d dimension pattern vectors. In other words, $\mathcal{X} \subseteq \mathbb{R}^d$. We assume that $\mathcal{X}$ is distributed between k classes.

$$\mathcal{X} = \bigcup_{i=0}^{k-1} \mathcal{X}_i$$

For our purpose, we split $\mathcal{X}$ in three parts: a learning set $\mathcal{L}$, a validation set $\mathcal{V}$, and a test set $\mathcal{T}$. Each of the k classes should be represented in each of these subsets.

We also consider two real numbers $T_R \in [0;1]$ and $T_C \in [0;1]$ that will be defined more precisely while designing our classifier.

2.2 Learning procedure

To begin, we perform a Principal Component Analysis (PCA) over the entire learning set. This kind of transformation is a classical tool in statistical data processing [2, 9, 3, 1]. The purpose is to find a new representation in which each pattern is as far as possible from the others. This space change is said to be a data decorrelation. It is conducted as following.

- The average learning pattern $\bar{\mathcal{L}}$ is computed over $\mathcal{L}$.
- Learning patterns are centered. That is, they are translated by $-\bar{\mathcal{L}}$.
- Covariance matrix for centered learning patterns is computed.
- Eigenvectors of covariance matrix are retrieved to form a new base in $\mathbb{R}^d$.
- Centered learning patterns are projected on this base.

At this point, a nearest neighbour classifier is implicitely built by saving $\bar{\mathcal{L}}$, the eigenbase and transformed learning patterns.

Then, validation patterns are translated by $-\bar{\mathcal{L}}$ and expressed in the eigenbase. Next task is to classify them using nearest neighbour procedure with euclidian distance and transformed learning patterns as references. Let $Conf\,(\mathcal{V})$ be the confusion matrix related to this classification. We normalize it column by column.

We examine each column j in $Conf\,(\mathcal{V})$ (with $j \in [\![0..k-1]\!]$). Value in cell (j,j) smaller than T_R means that less than $100 \times T_R\%$ of patterns labelled as j really belong to class j. Thus, T_R is the **minimal required accuracy** for each class during validation step.

Let j be a class with validation accuracy smaller than threshold T_R. We retrieve indexes $i \in [\![0..k-1]\!] \setminus \{j\}$ such as $Conf(i,j) > T_C$. Threshold T_C is the **maximal acceptable confusion** in validation step. We name I_j the set of i indexes found in that way. Classes with label in I_j are said to be **adherent** to j.

If $I_j \neq \emptyset$, we define the following new collections.

$$\mathcal{L}_{I_j} = \bigcup_{i \in I_j \cup \{j\}} \mathcal{X}_i \cap \mathcal{L} \text{ and } \mathcal{V}_{I_j} = \bigcup_{i \in I_j \cup \{j\}} \mathcal{X}_i \cap \mathcal{V}$$

where patterns are from original sets (*i.e.* before translation and base change).

For each non-trivial index set I_j, new PCA process is performed using $\mathcal{L}_{I_j}$ and $\mathcal{V}_{I_j}$ for learning and validation respectively. This is equivalent to consider a sub-problem involving only classes from $I_j \cup \{j\}$. Let us write $N_j = I_j \cup \{j\}$

If χ is the classifier built using $\mathcal{L}_{I_j}$, we define its **rank** as $r_\chi = |I_j| + 1 = |N_j|$ and χ is the **basic classifier** related to N_j.

Prediction over $\mathcal{V}_{I_j}$ is performed for χ and clusters of adherent classes are determined. These will lead to design new basic classifier.

2.3 Prediction

Prediction may be conducted in two different ways depending of the size of pattern set resulting in two different procedures.

Let $X \in \mathcal{T}$ be a pattern to label. We perform a classification *via* the maximum rank basic classifier (the first one built during learning step) and consider j, the predicted label. If $N_j = \emptyset$, the process ends. Otherwise, a new prediction involving basic classifier trained over classes of N_j is done.

The proposed classification procedure for a single test pattern does not admit any alternative. We now discuss the general case, where many vectors have to be labelled.

In intuitive approach, each pattern is processed separately using the precedent algorithm. This is a correct but (potentially) very time consuming way. Indeed, some basic classifiers may be loaded many times: transformed reference patterns, corresponding labels, average pattern, eigenbase have to be recalled, then discarded. To avoid such computation overload, we suggest a procedure based on a loop over isolated classifiers rather than over separate patterns.

Learning process has to be slightly adapted. Each basic classifier receives a special and non-ambiguous label, for instance, its order of construction during learning. Modifications for labelling algorithm are quite more subtle: each test pattern gets two temporary labels. One indicates the class prediction and the other refers to a classifier. At the begining of the process, all reference labels are set to point at the maximum rank classifier. Prediction labels are set to some default value.

We choose χ, the most frequently referred classifier with greatest rank (if there are many possible candidates, we choose the one with the smallest index). χ is used to perform prediction over all patterns that point at it. Such patterns are defined as **calling patterns** for χ and their set will be written $\mathcal{T}_\chi$.

Class labels are updated for all calling patterns, using predictions calculated *via* χ. For a given calling pattern, if its new class label belongs to an adherent class cluster regarding χ, then its reference label is updated to point at corresponding basic classifier. Otherwise, reference label is set to some absurd value, indicating that no further reclassification is needed.

Re-labelling is resumed with another frequently referred classifier until all reference labels are finally set to absurde value.

3. Remarks

One may quote that our cascade machine is conceptually close to a hierarchical classifier. When a pattern X is labelled as j, a set N_j of indexes indicates classes adherent to j. Assuming a non void N_j, X is re-classified. In other words, X is implicitely sorted to a super-class N_j. Thus, the cascade classifier process X *via* successive appointments to super-classes ordered by inclusion. This is typically a top-down hierarchical classification mechanism which could be related to decision tree paradigm [3, 1], except that each appointment is based on a metric analysis. Moreover, super-classes (*i.e.* clusters of adherent classes) are automatically defined during validation steps.

Local re-training and re-labelling mechanism are not totally new concepts. L. Prevost [8] proposed a two level architecture: a classifier is trained to separate all classes in a given problem. After a validation step, pairs of confusing classes are detected and specifically examined. Our cascade classifier can be viewed as a generalization of this approach without *a priori* knowledge (the number of classes in adherent clusters is free).

A significant part of previous works on classifier combination is focused in fact on classifier selection [5, 6]. Many machines (possibly from different kinds, or with different parameter values) are trained as concurrent experts. One of them is chosen in prediction step. Thus, several difficulties arise: designing many classifiers may be time consuming. Selecting an expert is not a trivial task. One must find appropriate heuristics according to the nature of the considered problem, the type and the distribution of data in the representation space, *etc.* That is, a meta-classifier has to be produced.

4. Experiments

We ran our classifier on various data sets. We present some results for the most interesting database: the MNIST handwritten digit corpus. We considere two kinds of results: global, or **raw accuracy** is defined as the ratio of correctly classified patterns over a test set. The **class accuracy** is the average value for accuracies computed in each class of patterns. This last measure is expected to be less sensitive to class distribution balance.

4.1 MNIST database

Modified NIST corpus, also called **MNIST** has been designed by Y. LeCun[1] *et al.* [7]. This database is a collection of 255 level grayscale handwritten digit images (examples in Fig. 1). Each sample image is 28×28 size.

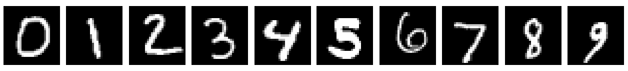

Figure 1. Examples of handwritten digit images in MNIST

Actually, **MNIST** is the union of two subsets from databases **SD-1** and **SD-3** proposed by the **NIST**[2] consortium (*National Institute of Standards and Technology*). **SD-3** was collected among Census Bureau employees, while **SD-1** was collected among high-school students. Hence, **SD-3** is easier to recognize than **SD-1**. To avoid influence of database choice for training or test on classifiers performances, patterns are mixed in **MNIST**.

In **SD-1**, 500 different writers produced $58\ 527$ digits. Patterns are shuffled. On the opposite, blocks of data from each writer appeared in sequence in **SD-3**. Since identities are available for **SD-1**, writers have been unscrambled and the dataset has been splitted in two parts: characters written by the first 250 writers went into a new training set. The remaining 250 writers were placed in the new test set. Thus two sets with nearly $30\ 000$ samples each were built. The new training set is completed with enough examples from **SD-3**, starting at pattern 0, to become a full set of $60\ 000$ training patterns. Similarly, the new test set is completed with **SD-3** examples starting at pattern $35\ 000$ to make a full set with $60\ 000$ patterns.

In most of the previous works involving **MNIST** database, only $10\ 000$ test patterns are used. To make comparisons possible, we chose to keep this restriction. We split training set in two balanced parts for learning and validation.

[1]http://yann.lecun.com/exdb/mnist/index.html
[2]http://www.nist.gov/

4.2 Dimensionnality reduction

To decrease computation load for prediction processes (during validation steps and separate test procedure), it is possible to perform a **dimensionnality reduction**. (Actually, this is one of the main motivations of PCA!) To achieve this, we only keep d' eigenvectors (with $d' < d$) while designing a basic classifier. Coordinate transformations will produce new pattern vectors of size d', more tractable.

To give an example, only 298 principal components are needed to express 90% of variance when performing PCA on the first $30\ 000$ training patterns. In other words, essential part of the information in the training base may be stored using only half-dimensional representation.

In practice, we set a threshold $\nu \in]0;1]$ corresponding to the minimal ratio of variance to be kept for analysis. For each basic classifier to be designed, eigenvectors extracted from learning pattern covariance matrix are ordered according decreasing eigenvalues. Let v_k (resp. V_k) be the k-th value (resp. vector) arranged in this way. We keep the n_ν first eigenvectors to form a base change matrix with

$$n_\nu = min \left\{ n \in [\![1..d]\!] \;\middle|\; \frac{\sum_{k=1}^{n} v_k}{\sum_{k=1}^{d} v_k} \geq \nu \right\}$$

In other words, n_ν is the minimal number of principal components to be kept to capture $100\nu\%$ of learning data variance. It may vary from a rank to another or from a dataset to another.

Results for experiments tuning ν value are presented in Table 1. Minimal required accuracy on validation set for basic classifiers is chosen to be $T_R = 0.95$, maximal acceptable confusion ratio for each class is fixed at $T_C = 0.01$.

Table 1. Effect of variance ratio on accuracy over test patterns. Parameters are set to $T_R = 0.95$ and $T_C = 0.01$.

Learning	Validation	ν	Accuracy (%)	
			raw	class
30 000	30 000	0.80	96.16	96.21
30 000	30 000	0.85	96.35	96.40
30 000	30 000	0.90	96.26	96.32
30 000	30 000	0.95	96.07	96.13
30 000	30 000	1.00	96.13	96.18

In first experiment, we used half of the patterns in training dataset for learning, and the others for validation. We chose a variance ratio of $\nu = 0.8$ to decrease computation load. Accuracy on test set is 96.16%, which is far better than performances obtained by linear classifiers (results for experiments using different classifiers are reported in [7]).

Examination of the global confusion matrix (see Table 2) for test patterns leads us to quote some trends. We establish good recognition for digits of classes *zero* and *one*. Confusion rates are very low (all values under 0.5%). On the opposite, classes *four* and *nine* are very adhering. Similar remark may be issued for classes *thee* and *five*. These confusions may be explained by strong morphological similarity for handwritten digits (see example in Fig. 2).

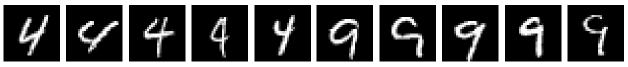

Figure 2. Examples of handwritten digit images in MNIST. Five left (resp. right) images show distorted “four” (resp. “nine”).

4.3 Effects of thresholds

Two other parameters may potentially influence our cascade classifier’s generalization capacity: thresholds T_R and T_C used during validation steps.

To simplify, we chose to focus on T_R only. T_C is set *via* the heuristic relation

$$T_C = \frac{1 - T_R}{k - 1}$$

where k is the number of classes in the global classification problem.

We realised experiments with different values for T_R. We quote that variations of T_R between a wide range from

Table 2. Confusion matrix for $\nu = 0.8$

98.77	0.10	0.20	0.0	0.0	0.10	0.41	0.10	0.20	0.10
0.0	99.47	0.26	0.26	0.0	0.0	0.0	0.0	0.0	0.0
0.77	0.29	95.93	0.68	0.58	0.0	0.19	1.55	0.0	0.0
0.20	0.10	0.40	96.34	0.0	1.58	0.0	0.69	0.10	0.59
0.0	0.51	0.0	0.0	96.23	0.0	0.41	0.20	0.10	2.54
0.22	0.0	0.11	2.47	0.11	95.29	0.67	0.34	0.67	0.11
0.63	0.42	0.21	0.0	0.42	0.21	98.12	0.0	0.0	0.0
0.0	1.94	0.78	0.19	0.39	0.0	0.0	95.82	0.10	0.78
0.41	0.20	0.92	2.87	0.10	1.44	0.72	0.92	91.68	0.72
0.30	0.40	0.10	0.69	2.38	0.49	0.10	1.09	0.49	93.95

0 to 0.90 have no effect on accuracy for test set if we chose $\nu = 1$. Results reported in Table 3 show this phenomenon. They also give an illustration of overtraining effect when setting T_R to excessive values ($T_R = 0.95$ and $T_R = 1.0$): constraints over accuracy in validation steps are so strong that generalisation is penalized.

To confirm our assertions, we tried different value of ν parameter. Results are reported in Table 4. Class accuracies are collected and presented as main results. Raw accuracies are also indicated as secondary indicators in brackets. Once again, overtraining appears for T_R around 0.95.

5 Further work

We proposed a cascade classifier based on confusion matrix analysis [4]. This system has been implemented and experiments show encouraging results. Our tool is still under development and many improvements are possible.

Source code may be optimized. We were mainly interested in classifier’s behavior and produced a software to run

Table 3. Set of experiments tuning minimal acceptable validation accuracy. Variance ration set to $\nu = 1$

Learning	Validation	T_R	Accuracy (%)	
			raw	class
30 000	30 000	0.05	96.13	96.18
30 000	30 000	...	...	...
30 000	30 000	0.90	96.13	96.18
30 000	30 000	0.95	95.66	95.74
30 000	30 000	1.00	94.66	94.73

our experiments, without questioning about the efficiency of **data structures** or the optimality of the implemented procedures. The predicted computational complexity is likely to be decreased using implementation tricks (look-up tables, loop optimization, *etc*.)

Heuristic rules are still to be found for **parameter optimization**. For instance, minimal validation accuracy threshold may be more precisely determined using considerations on dataset size, number of classes to be separated, rank of basic classifier (compared to global problem rank), *etc.* Introducing more parameters, joint analysis of effects will be preferred. Analysis of variance (ANOVA) and design of experiments will help us in such task.

The cascade classifer concept may be reused with **other architectures**. We built basic classifiers based on principal component analysis and nearest neighbour labelling because of the conceptual simplicity of such choice. Slight adaptation in our code allows us to introduce n nearest neighbours labelling (with $n \geq 1$) which is expected to be more precise. Experiments are under progress to confirm it.

Our current basic classifiers are based on PCA and neighbourhood attraction. In even further perspective, we plan to replace them by Support Vector Machines (SVM), based on inter-class margin optimization. Linear SVM will be tested before considering more complexe classifiers (*e.g.* involving kernel functions).

Table 4. Set of experiments tuning validation threshold and variance ratio. Class (resp. raw) performances on test set are presented as main (resp. secondary) figures.

	ν				
T_R	0.80	0.85	0.90	0.95	1.00
0.80	96.08 (96.14)	96.20 (96.25)	96.17 (96.23)	96.04 (96.10)	96.09 (96.14)
0.85	=	=	=	=	=
0.90	=	=	=	=	=
0.95	95.79 (95.86)	95.77 (95.84)	95.81 (95.89)	95.66 (95.74)	95.66 (95.74)
1.00	..	..	..	..	94.66 (94.73)

References

[1] J.-M. Bouroche and G. Saporta. *L'Analyse des Données.* Que sais-je? Presses Universitaires de France (PUF), Vendôme (France), February 2002.

[2] J. de Lagarde. *Initiation à l'Analyse des Données.* Dunod, Saint-Etienne (France), 1983.

[3] R. Duda, P. Hart, and D. Stork. *Pattern Classifi cation.* Wiley Interscience, 2001.

[4] J. Duong, R. Sabourin, and H. Emptoz. Proposition d'un classifi cateur en cascade : application à la reconnaissance de polices de caractères rares. In *Actes du* 8^e *Colloque International Francophone sur l'Ecrit et le Document (CIFED'2004)*, pages 231–236, La Rochelle (France), 21-25 June 2004.

[5] G. Giacinto and F. Roli. Automatic design of multiple classifi er systems by unsupervised learning. In *Proceedings of the* 1^{st} *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, September 16-18 1999. Lecture Notes in Artifi cial Intelligence, vol. 1715, pp. 131-143.

[6] G. Giacinto and F. Roli. Dynamic classifi er selection. In *Proceedings of the* 1^{st} *International Workshop on Multiple Classifi er Systems*, Cagliari (Italy), June 21-23 2000. Lecture Notes in Computer Science (LNCS), vol. 1857, pp. 177-189.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] L. Prevost, C. Michel-Sendis, L. Oudot, and M. Milgram. Combining model-based and discriminative classifi ers : application to handwritten character recognition. In *Proceedings of the* 7^{th} *International Conference on Document Analysis and Recognition (ICDAR'2003)*, pages 31–35, Edinburgh (Scotland, United Kingdom), August 3-6 2003.

[9] G. Saporta. *Probabilités, analyse des données et statistique.* Technip, Paris (France), 1990.