Comparing Combination Rules of Pairwise Neural Networks Classifiers

Olivier Lezoray, Hubert Cardot

Comparing combination rules of pairwise neural networks classifiers

Olivier Lézoray (olivier.lezoray@unicaen.fr)
Université de Caen, GREYC UMR CNRS 6072, 6 Bd. Maréchal Juin, Caen F-14050, France

Hubert Cardot (hubert.cardot@univ-tours.fr)
Université Francois-Rabelais de Tours, Laboratoire d'Informatique EA 2101, 64 Avenue Jean Portalis, Tours, F-37200, France

Abstract. A decomposition approach to multiclass classification problems consists in decomposing a multiclass problem into a set of binary ones. Decomposition splits the complete multiclass problem into a set of smaller classification problems involving only two classes (binary classification: dichotomies). With a decomposition, one has to define a recombination which recomposes the outputs of the dichotomizers in order to solve the original multiclass problem. There are several approaches to the decomposition, the most famous ones being one-againt-all and one-against-one also called pairwise. In this paper, we focus on pairwise decomposition approach to multiclass classification with neural networks as the base learner for the dichotomies. We are primarily interested in the different possible ways to perform the so-called recombination (or decoding). We review standard methods used to decode the decomposition generated by a one-against-one approach. New decoding methods are proposed and compared to standard methods. A stacking decoding is also proposed which consists in replacing the whole decoding or a part of it by a trainable classifier to arbiter among the conflicting predictions of the pairwise classifiers. Proposed methods try to cope with the main problem while using pairwise decomposition: the use of irrelevant classifiers. Substantial gain is obtained on all datasets used in the experiments. Based on the above, we provide future research directions which consider the recombination problem as an ensemble method.

Keywords: Pairwise classifier; combination; stacking.

1. Introduction

Since the advent of data mining in information management systems, the applications of multiclass pattern recognition has covered a very wide range including image or text categorization, object classification, speech recognition. Multiclass pattern recognition aims at building a function F that maps the input feature space to an output space of more than two classes. Each example (x, y) consists of an instance $x \in X$ and a label $y \in \{1, ..., K\}$ where X is the feature vector input space and K the number of classes to be discriminated. A general classifier can be considered as a mapping F from instances to labels $F : X \rightarrow \{1, ..., K\}$. There are two ways for a classifier to solve multiclass problems: (1)

consider all the data in one optimization problem, (2) construct several binary classifiers and combine them. The first approach formulates the multiclass problem into one single optimization problem (all-at-once). However, the number of samples is the main factor that contributes to the time complexity for training the classifier. Therefore, algorithms of the first category are significantly slower than ones that include several binary classifiers where each classifier classifies only a small portion of the data [11, 12, 23, 34]. Moreover, muticlass classification is intrinsically harder than binary classification because the classification algorithm has to learn to construct altogether a high number of separation boundaries whereas binary classifiers have to determine only one appropriate decision function. Therefore, the machine learning community has devoted considerable attention to the latter type of approach the principles of which are the following. A decomposition splits a complete multiclass problem into a set of less complex and independent two-class problems (dichotomies) and a recombination (a decoding) recomposes the outputs of dichotomizers in order to solve the original polychotomy [11, 21]. A binary classifier having to face with the classification of data using examples as positive ones and the others as negative ones, there are two schemes to construct several binary classifiers for multiclass problems [31]. The most traditional scheme [36] splits the original problem into a series of binary problems (one for each class) where the i^{th} classifier is trained while labeling all the samples in the i^{th} class as positive and the rest as negative. This technique is called one-against-all since each classifier separates one class from all the others. The drawbacks of this approach is that each binary classifier has to see all the training database instead of a reduced version of it and the training data can be unbalanced which can distort each binary classifier. The second scheme constructs $K \times (K-1)/2$ classifiers using all the pairwise combinations of the K classes [13, 34, 39]. This technique is called one-against-one (pairwise). The latter approach is very interesting since the binary decision is not only learned on fewer training examples but the decision function of each binary problem can be considerably simpler than in the case of one-against-all [11, 12, 27]. The major advantage of the pairwise approach is that it provides redundancy which can lead to better generalization abilities. Pairwise classification has been tried in the areas of statistics [2, 10], neural networks [19, 27, 28, 34], support vector machines [13, 15, 21] and others (bioinformatics [7, 16], speech recognition [17], pathology [25]). Typically, this technique learns more accurate concepts than the more commonly used one-against-all classification method with lower computational complexity (see in [12]). Furthermore, the advantage of pairwise classification increases for computationally expensive (super-linear) learning algorithms. The

reason is that expensive learning algorithms learn many small problems much faster than a few large problems. Pairwise classifiers can also be tuned so as to use a specific subset of features which can be different for each pairwise classification problem.
Beside choosing the way to decompose the problem, one also needs to devise a strategy for combining the outputs of binary classifiers and provide a final prediction. Whatever the used decomposition (one-against-all and one-against-one), if a simple voting strategy is used, there can be inconsistent regions (less for one-against-one but one still remains, see [39]). The problem of combining binary classifiers has therefore been extensively studied and a lot of combining schemes have been proposed but many researchers reported opposing views to which scheme is better in terms of accuracy and speed [1, 11, 12, 13, 15, 26, 29, 30, 31, 33, 36, 38, 39]. Speed issues depend primarily on the different implementations of the basic binary classifier and accuracy issues depend on the nature of the basic learner, the data set and how the basic classifiers are well tuned to achieve maximum performance [36]. Literature being inconclusive (recent papers still claim that there is non clear superiority over the methods [8]), the best method for combining binary classifiers is an important research issue that remains open. In this paper, we propose to compare several classical combining schemes using Multi Layer Perceptrons (MLP) as the base learner. Moreover, we also propose new combining schemes which outperform the classical ones. We consider only the one-against-one formulation to proceed to the construction of binary neural networks. In Section 2, we present the binary neural networks we used. In section 3, we discuss how to combine binary neural networks. In Section 4, we demonstrate the effectiveness of the proposed combining methods by computer experiments. Last Section draws a conclusion and future research directions which consider the recombination problem as an ensemble method.

2. Binary Neural Networks

Since we consider the one-against-one decomposition scheme, for a classification problem with K classes, a set of neural networks is built; each one being in charge of separating elements from two distinct classes. The set of different classes is denoted by $C = \{C_1, C_2, ..., C_K\}$ with $|C| = K$. The set of Binary Neural Networks (BNN) is given by $\mathcal{A} = \{\mathcal{R}_{C_1,C_2}; \cdots; \mathcal{R}_{C_{K-1},C_K}\}$. In the rest of the paper, we will equally use the notation $\mathcal{R}_{C_i,C_j}$ or $\mathcal{R}_{i,j}$ for sake of simplicity. The global training dataset containing patterns of all the different classes is denoted by D_{Train}. The latter is divided in several subsets for each neural network.

$D_{Train}(C_i, C_j)$ is the dataset that corresponds to the neural network which differentiates the classes C_i and C_j and contains patterns of only those two classes. Then, the initial training data ($D_{Train}(C_i, C_j)$) associated to each neural network is split into two subsets: a learning set ($D_{Learn}(C_i, C_j)$) and a validation set ($D_{Valid}(C_i, C_j)$). The latter consists in 20% of $D_{Train}(C_i, C_j)$ and the learning set in 80% of $D_{Train}(C_i, C_j)$. The learning of a neural network is performed on $D_{Learn}(C_i, C_j)$ and the $D_{Valid}(C_i, C_j)$ validation set is used to evaluate the classification rate of the network during the training. Therefore, the validation set is not used to learn the weights of neural networks, but only to tune the hyper-parameters (number of neurons, number of iterations, etc.). The structure of the neural networks used is the following one: a layer of inputs containing as many neurons as the number of features associated with the object to be classified, a hidden layer containing a variable number of neurons and one output neuron. The value of the output neuron is in the interval $[-1, 1]$. According to the sign of the result associated with this single neuron, an object is classified in one of the two classes that the network separates. The neural networks used are very simple: only one hidden layer, only one neuron of output. This has several advantages [4, 23, 26]. The simplicity of the task associated to each neural network simplifies the convergence of the training as well as the search for a simple structure. Therefore, an automatic method is used to find the number of hidden neurons that gives the best classification rate [23, 3, 22]. The output value provided by a BNN when a sample data x has to be classified is denoted by $O(x, \mathcal{R}_{C_i,C_j})$. From this output, x can be classified as C_i or C_j according to the sign of the output: x is considered as of class C_i if $O(x, \mathcal{R}_{C_i,C_j}) \geq 0$ and C_j otherwise. The output can therefore be directly used as an estimate for the class memberships. However, it might be more suitable to have pairwise class probability estimates which can be obtained by [34] $r_{ij}(x) = (O(x, \mathcal{R}_{C_i,C_j})+1)/2$ and $r_{ji}(x) = 1-r_{ij}(x)$. $r_{ij}(x)$ estimates the pairwise posterior probability of the input vector x to belong to the class i and $r_{ji}(x)$ to the class j (according to the single BNN $\mathcal{R}_{C_i,C_j}$).

3. Combining binary classifiers

Constructing multiclass classifiers from a pairwise decomposition consists in combining the $B = (K \times (K-1)/2)$ pairwise classifiers outputs. Each binary classifier is a mapping $f_b : X \to \mathbb{R}$ with $b \in \{1, ..., B\}$. A vector $f(x) = (f_1(x), ..., f_B(x))$ is constructed from the outputs of the binary classifiers. A combination rule g can then be applied to combine

all the outputs $f(x)$ using a function $g : \mathbb{R}^B \rightarrow \mathbb{R}^K$ which couples the estimates of each binary classifier in order to obtain class membership estimates (which can be probabilities) for the multiclass problem. Once the class memberships μ have been estimated as

$$g(f(x)) = g(f_1(x), ..., f_B(x)) = (\mu(C_1|x), ..., \mu(C_K|x)),$$

a final selection scheme is used to choose the winner class. This is done as a mapping $h : \mathbb{R}^K \rightarrow \{1, ..., K\}$. The whole K-ary classifier combining all the binary classifiers scores (obtained by pairwise decomposition) is denoted by $F(x) = h(g(f(x)))$ where h is the selection scheme function applied to select the winner class and g the combination rule. $h \circ g$ defines the complete decoding scheme needed to perform multiclass classification from binary pairwise classifiers.

3.1. Standard Decoding

In this Section, we review the standard decoding schemes based on class memberships estimation and propose a new one.

3.1.1. *Majority vote*

The most commonly used combination rule is probably the Majority Vote (MV) one. With this combination rule [10], each class receives votes from individual classifiers. The membership estimates correspond to the number of votes received by each class: $\mu(C_i|x) = \Sigma_j V(r_{ij}(x) \geq 0.5)$ with $V(x) = 1$ if x is true and 0 otherwise. The chosen class is the one which receives the largest number of votes and $h = argmax$.

3.1.2. *Hastie*

A way to obtain class membership estimates from the pairwise probability estimates $r_{ij}(x)$ has been proposed by Hastie [13]. To combine all the estimates of the BNN, we would like to obtain a set of class membership probabilities $p_i(x) = P(C_i|x)$. The r_{ij} estimate class conditional probabilities and are related to the p_i according to

$$r_{ij}(x) = P(C_i|x, C_i \vee C_j) = \frac{p_i(x)}{p_i(x) + p_j(x)}$$

In order to find the best approximation $r'_{ij}(x) = \frac{\mu_i(x)}{\mu_i(x)+\mu_j(x)}$, the algorithm starts with $\mu_i(x) = \frac{2\Sigma_{j \neq i} r_{ij}(x)}{K(K-1)}$ and computes the corresponding

$r'_{ij}(x)$. The $\mu_i(x)$ are obtained by minimizing the average Kullback-Leibler distance between $r_{ij}(x)$ and $r'_{ij}(x)$:

$$\mathcal{L}(\mu) = \sum_{i<j} n_{ij} \left(r_{ij} \log \frac{r_{ij}}{r'_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - r'_{ij}} \right)$$

where n_{ij} is the number of examples of classes i and j in the training set. The associated equations are: $\sum_{j \neq i} n_{ij} r'_{ij} = \sum_{j \neq i} n_{ij} r_{ij}$ subject to $\sum_{k=1}^{K} \mu_k = 1$. At convergence, we have the class membership estimates with $\mu(C_i|x) = \mu_i(x)$. The winner class is considered as the most likely one and $h = argmax$.

3.1.3. *Price*

Another approach for estimating the class memberships has been proposed by Price [34]. It is based on the fact that neural networks trained to minimize a MSE cost function estimate posterior probabilities. A BNN with sigmoidal transfer function can therefore estimate the posterior probabilities for the two classes (previously denoted by $r_{ij}(x)$ and $r_{ji}(x)$). One can then obtain the final expression of the class membership estimates by $\mu(C_i|x) = \frac{1}{\sum_{j \neq i} \frac{1}{r_{ij}(x)} - (K-2)}$. As for the Hastie combination rule, we have $h = argmax$.

3.1.4. *Loss Based Decoding (LBD)*

Another interesting combination rule is based on Error Correcting Output Codes (ECOC) [1, 5, 6]. Different classes are represented by output codes in the output vector space and the prediction of classification is based on the fact that the chosen class has its output code close to the output one of the example to be classified. It has been introduced to combine the outputs of binary Support Vector Machines. For a problem with K classes, it creates a matrix $M \in \{-1, 0, 1\}^{K \times B}$. For a $K = 3$ classification problem, the matrix is defined as:

	f_1	f_2	f_3
C_1	+1	+1	0
C_2	-1	0	+1
C_3	0	-1	-1

A column in the matrix M corresponds to a binary classifier $\mathcal{R}_{i,j}$ and a row corresponds to a class. For instance, the first column corresponds to the classifier $\mathcal{R}_{C_1,C_2}$. Therefore, to each class is assigned

a binary code vector of length K which makes up the rows of the code matrix. Combining all the binary classifiers to estimate the class memberships consists in comparing the matrix rows with the classifiers outputs expressed by $\mu(C_k|x) = \sum_{b=1}^{B} L(M(k,b).f_b(x))$. This provides the distortion between the vector of the BNN outputs $f(x)$ and the row $M(k,\cdot)$. For this combination rule, the outputs of the binary classifiers $O(x, \mathcal{R}_{ij})$ are directly used and not the $r_{ij}(x)$. The original approach to ECOC predicts the class whose corresponding row vector has minimum Hamming distance to the vector of 0/1 predictions obtained from the classifiers [6]. However, the accuracy can be improved by using Loss Based decoding (LBD) and we have therefore used $L(z) = exp(-z)$. This means using the outputs of the binary classifiers rather than their hard 0/1 predictions [1]. For LBD decoding schemes $h = argmin$ since this leads to finding the row being the most similar to the classifiers outputs [17, 18] (nearest neighbor decoding). Some works consider learning the weights of the code matrix by solving an optimization problem [16] but we do not consider that issue in this paper.

3.1.5. *Min-Max*

For all the previous decoding schemes, the probability estimates of the classifiers obtained by the combination rule g are used to assign to an input pattern the class with the maximal output. Combining all the pairwise classifiers can lead to bad results since if the input x is of class C_i, there are only $(K-1)$ relevant classifiers among the $(K \times (K-1))/2$ which have seen the class C_i and the remaining $((K-1) \times (K-2))/2$ irrelevant classifiers have never seen inputs from class C_i. While classifying an input, one wishes that relevant classifiers will provide coherent informations to cope with all the irrelevant ones. Moreira has proposed [30] to estimate the relevance of a binary classifier with correcting classifiers which separate each pair of classes from all the other classes; correcting classifiers being used to weight the pairwise probabilities. This approach is however very expensive and the advantages of the pure pairwise approach are lost. To try to alleviate this problem, we propose to get the minimum value of $r_{ij}(x)$ for each class C_i : $\mu(C_i|x) = \min_{j}^{K-1} r_{ij}(x)$. Finally, we select the class which maximizes this minimum value: $h = argmax$. The principle of this method consist in choosing the candidate class whose probability is less bad than that for all other candidate classes. The intuitive idea behind is that having a high pairwise probability for a particular pair of classes does not imply a strong decision towards this class because

of irrelevant classifiers. However, it can be rejected if the probability is low.

3.2. Elimination Decoding

Another decoding scheme is the elimination decoding one. This decoder was originally described by Kressel [21] and reintroduced by Platt [33] where it was called Directed Acyclic Graph (DAG). One strong argument for using DAG is that it resolves the problem of unclassifiable regions for pairwise classification. The elimination decoding is nothing more than a decision-tree based pairwise classification. The nodes of this tree are binary classifiers. To perform a multiclass classification, an example is presented to each node and follows a path in the tree by eliminating a class at each node. The set of binary classifiers $\mathcal{A}_0 = \{\mathcal{R}_{1,2}, ..., \mathcal{R}_{K-1,K}\}$ contains all the binary classifiers and the set $E_0 = \{C_1, ..., C_K\}$ contains all the candidate winner classes. Elimination decoding operates iteratively. At each iteration $t = \{1, ..., K-1\}$, a binary classifier performs a decision and one class C_k is eliminated. The size of E_t is then decreased by one and all the classifiers discriminating C_k in $\mathcal{A}_t$ are eliminated [18]: $\mathcal{A}_{t+1} = \mathcal{A}_t - \{\mathcal{R}_{i,j} : i = C_k \vee j = C_k\}$. The set $\mathcal{A}_{K-2}$ contains only one binary classifier which determines the winner class. Several problems occur however when using DAGs. First of all, the choice of the winner class depends on the sequence of binary classifiers in nodes which affects the reliability of the algorithm. Moreover, the correct class to be predicted is more or less advantaged according to its distance to the root node (higher risk of being rejected in the nodes near the root). Secondly, since there are a lot of classifiers which are irrelevant for a given classification, using these classifiers can cause severe defects. To overcome this problem, several authors have proposed to use an Adaptive DAG (called ADAG) by optimizing its structure. However, the generalization ability still depends on the structure of the tree [26, 4, 20, 32, 40]. We propose a new elimination decoding which takes into account all the outputs of the binary classifiers. When using a classical decoding scheme without elimination, one selects the class with the largest probability ($h = argmax$). In the case of elimination, we want to eliminate the least credible class and this comes back to eliminate the class having the minimum of probability. The class to be eliminated is deduced from the class membership estimates: $C_k = h(g(f(x)))$. g can be any of the previous combination rules (Majority Vote, Hastie, Price, LBD, Max-Min) and since the method eliminates the least probable class at each iteration, we have $h = argmin$. At the iteration t, the number of candidate classes is $|E_t| = (K - t)$ and the number of binary classifiers to be combined is

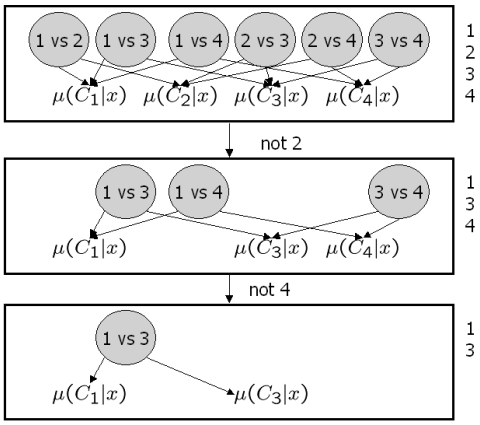

Figure 1. Illustration of the proposed new elimination decoding. At each step, a class is eliminated. The remaining candidate classes are shown on the right of boxes designing a step of the algorithm.

$|\mathcal{A}_t| = (K-t)(K-t-1)/2$. This new elimination decoding is different of the Direct Acyclic Graph [33] since at each iteration, all the outputs of candidate binary classifiers are combined to determine the class to be eliminated. Indeed, with a DAG, only one classifier output is used for eliminating a class at each iteration. The proposed elimination decoding can be summarized by Algorithm 1. Figure 1 presents an illustration of the algorithm for a 4-class problem.

Algorithm 1 The proposed new elimination decoding.

$t \leftarrow 0$, $\mathcal{A}_t \leftarrow \{\mathcal{R}_{0,1}, ..., \mathcal{R}_{K-1,K}\}$, $E_t \leftarrow \{C_1, ..., C_K\}$
while $|E_t| \neq 1$ **do**
$C_k \leftarrow h(g(f(x)))$
$E_{t+1} \leftarrow E_t - \{C_k\}$
$\mathcal{A}_{t+1} \leftarrow \mathcal{A}_t - \{\mathcal{R}_{i,j} : i = C_k \vee j = C_k\}$
$t \leftarrow t + 1$
endwhile
$F(x) = E_t$

3.3. Stacking Decoding

The combination of the class membership estimates can be performed via a separate trainable classifier [17, 37]. This method is referred to as stacking [41]. Very few works consider this issue for multiclass classification. Klautau [17] and Mayoraz [29] both use a neural network to stack the predictions provided by one-versus-all binary classifiers. Savicky combines binary decision trees with a meta decision tree at

Table I. Classification cases to illustrate the advantages of stacking decoding.

Classifier			Original
C_1vsC_2	C_1vsC_3	C_2vsC_3	class
C_1	C_1	C_2	C_1
C_2	C_3	C_2	C_2
C_1	C_3	C_3	C_1
...	...	...	...
C_2	C_3	C_3	C_3

the stacking level [37]. This approach seems more suitable than all the previous ones for the following reason. As said before, combining classifiers which have never seen instances from one same class during the training phase results in combining different information sources. The combination of these ignorant classifiers (a binary classifier has seen only two classes among K) with respect to the others can therefore result in almost random classification. Indeed, decoding methods such as voting rely on the assumption that the relevant classifiers mainly predict the correct class and provide more votes to the true class than the irrelevant classifiers to any other class. However, if some of the relevant classifiers predict wrong classes, the final classification can be also wrong. Since we cannot predict the behavior of irrelevant classifiers, more sophisticated decoding schemes are needed. The potential gain of stacking for decoding is evident and it can lead to correct predictions where other methods would fail [37, 42]. Table I illustrates this fact and presents classification results for a 3-class problem (from [37]). With a standard decoding such as voting, the decoding will always fail to predict the correct class of the third sample. Indeed, the classifier C_1vsC_3 makes a mistakes (it predicts class C_3) and the classifier C_2vsC_3, which is irrelevant for this classification (the input sample is of class C_1), predicts the input sample as of class C_3. This case illustrates all the problems we have to face with when using binary classifiers: some classifiers are irrelevant and the relevant classifiers can make mistakes. Simple standard decoding schemes cannot cope with such cases and one has to learn to recognize such ones: a Meta Classifier can be used to that aim. We therefore propose two approaches to stacking decoding.

Stacking consists in combining the outputs of different classifiers to feed a trainable Meta Classifier. With a pairwise decomposition approach to classification, two approaches to stacking can be consid-

ered. The first approach to stacking is to learn on all the probabilities estimated by any standard decoding. In standard decoding, the whole K-ary classifier combining all the binary classifiers scores (obtained by pairwise decomposition) is denoted by $F(x) = h(g(f(x)))$. One can replace h by a trainable classifier to cope with irrelevant predictions provided by by some pairwise classifiers. In that case, each feature vector x_i of the training set D_{Train} is used to feed all the binary classifiers and class membership estimates are obtained with a combination rule g. A new example $(\tilde{x_i}, y_i) = (g(f(x_i)), y_i)$ is then obtained. The dimension of $\tilde{x_i}$ is equal to the number of classes and y_i is the known class of x_i. A new training set $D'_{Train} = \{(\tilde{x_1}, y_1), ..., (\tilde{x_N}, y_N)\} = \{(g(f(x_1)), y_1), ..., (g(f(x_N)), y_N)\}$ is generated from the class membership estimates. The new training data set is used to train a Meta Classifier which predicts the final class by $F(x) = h(g(f(x)))$. The function h designs the Meta Classifier to be used. Using a Meta Classifier which performs stacking on the class membership estimates is close to learn the weights of LBD decoding matrix [16].
The second approach to stacking is to learn on all the predictions of the binary classifiers [24]. To that aim, a trainable classifier is used with as training input the output vector $f(x)$ of all the binary classifiers. For this approach to stacking, new examples of the meta training set are $(\tilde{x_i}, y_i) = (f(x_i), y_i)$. In that case, the function $h \circ g$ designs the Meta Classifier to be used. The $D'_{Train} = \{(f(x_1), y_1), ..., (f(x_N), y_N)\}$ database generated by all the binary classifiers provides valuable information about the possible misleading predictions caused by the irrelevant classifiers.

4. Experimental results

This Section presents an experimental comparison of the ways to combine binary neural networks according to different combining rules. The databases for which results will be presented here are data bases coming from the Machine Learning Data Repository of the University of California at Irvine (UCI) [14]. Table II describes the different databases showing the variety of training data set sizes ($|D_{Train}|$), the number of classes (K), the number of neural networks (B) and the dimensionality of the data input ($|x|$). The tests are performed on a data set (D_{Test}) independent of the training set.

Tables III, IV and V presents all the classification rates obtained on D_{Test} for the different combining rules (best rates bold faced for each decoding rule family). For the standard decoding (Table III), the results are homogeneous, except for Hastie and Price methods which

Table II. Data bases used for the tests.

Database	K	$\|D_{Train}\|$	$\|D_{Test}\|$	$\|x\|$	B
Iris	3	120	30	4	3
Wine	3	144	34	13	3
Vehicle	4	679	167	18	6
PageBlocks	5	4382	1091	10	10
SatImage	6	4435	2000	36	15
Shuttle	7	43500	14500	9	21
PenDigits	10	7494	3498	16	45
OptDigits	10	3065	760	64	45
Letter	26	16000	4000	16	325

Table III. Classification rates of one-versus-all and one-against-one standard decoding.

	MLP	MV	Hastie	Price	LBD	MinMax
Iris	**70.00**	**70.00**	63.33	63.33	**70.00**	**70.00**
Wine	**97.06**	**97.06**	**97.06**	**97.06**	**97.06**	**97.06**
Vehicle	66.67	69.46	68.26	**70.06**	69.46	69.46
PageBlocks	84.80	88.27	42.35	46.29	**88.45**	88.36
SatImage	80.00	77.90	**80.10**	79.90	**80.10**	78.35
Shuttle	79.15	95.70	95.01	95.18	**95.82**	95.57
Pendigits	83.82	89.17	82.19	88.42	89.05	**89.22**
Optdigits	90.39	91.70	81.69	87.62	90.91	**91.83**
Letter	62.45	78.37	65.50	71.57	**78.52**	77.72

perform significantly worse on several datasets. As expected from the literature, LBD decoding performs very well and provides results always better than the Majority Vote. One thing to point out with LBD is that it can be a robust combining method as long as the errors of the binary classifiers are not correlated [30, 29]. For this purpose, all the dichotomies must be as distinct as possible, using well-tuned binary classifiers does the matter and explains why LBD works well in our study. When LBD is not best combining method, best results are obtained with the proposed Min-Max method which confirms the intuitive idea that in some cases the irrelevant classifiers have strong influence

on the final decision. Another advantage of the Min-Max method is its simplicity. One can therefore say that even if the results are very mixed, two standard decoding schemes can be retained as the best ones: LBD and the proposed Min-Max method.

Table IV. Classification rates of one-against-one elimination decoding.

	ADAG	MV	Hastie	Price	LBD	MaxMin
Iris	**70.00**	**70.00**	**70.00**	**70.00**	**70.00**	**70.00**
Wine	**97.06**	**97.06**	**97.06**	**97.06**	**97.06**	**97.06**
Vehicle	**69.46**	68.26	68.86	68.86	68.86	**69.46**
PageBlocks	**88.45**	88.36	88.36	**88.45**	**88.45**	88.36
SatImage	78.05	77.85	78.20	78.10	78.20	**78.35**
Shuttle	95.64	95.66	95.63	95.62	**95.76**	95.59
Pendigits	88.99	89.19	89.14	89.11	89.17	**89.28**
Optdigits	88.67	91.04	91.17	91.17	91.17	**91.70**
Letter	77.64	78.44	77.54	77.64	**78.47**	77.67

If we now have a look at the results obtained with the proposed elimination decoding method (Table IV), the first interesting thing is that the results look much more homogeneous between the different combining rules. As it was noted by Platt with DAGs, using an iterative elimination method reduces the error bound [33] since it avoids the problem of irrelevant regions of classification. Table IV presents a comparison for the proposed elimination decoding with an improved ADAG (a Graph of Neural Networks, see in [4, 26] for details). It can be noted that the elimination method we propose performs in general better than classical DAGs [15] even of improved structure; proving that using an elimination method based on combining rules is a more robust method than one based on a decision tree of binary classifiers, without the problem of optimizing the structure of the tree. This is all the more interesting since our elimination decoding reduces the error bound whatever the combining rule. As for the standard decoding method, the two best combining rules are LBD and Max-Min (since we perform an elimination we do not have Min-Max: at each iteration the class minimizing the highest pairwise probability is eliminated). Elimination decoding is however in average less good than standard

decoding and it has be preferred only when the latter does not perform well (e.g. for Hastie and Price standard decoding on the PageBlocks data base, see Table III). The inner limitation of elimination decoding is related to the problem of irrelevant classifiers which can predict wrong class with high confidence resulting in their early elimination.

Table V. Classification rates of stacking decoding with as input: class membership estimates by standard decoding (rows 1-4) or all the pairwise predictions (rows 5-6).

	Hastie	Price	LBD	MinMax	C4.5 (HP)	C4.5 (SP)
Iris	93.30	90.00	**96.70**	90.00	70.00	90.00
Wine	**100.00**	**100.00**	**100.00**	**100.00**	97.10	**100.00**
Vehicle	74.90	71.90	71.30	72.50	68.30	**75.40**
PageBlocks	91.30	**92.70**	91.20	92.90	89.60	92.30
SatImage	84.30	84.40	84.50	**85.60**	79.60	85.10
Shuttle	**99.80**	**99.80**	99.70	**99.80**	96.70	**99.80**
Pendigits	91.60	92.60	92.00	92.70	92.10	**93.10**
Optdigits	89.20	89.90	88.90	92.50	90.50	**93.02**
Letter	80.40	80.40	80.20	80.70	**81.00**	**81.00**

Finally, we analyze the results of our two proposed approaches to stacking decoding (Table V). We used decision trees (C4.5 [35]) Meta Classifiers to perform stacking. On all stacking datasets, a 10-fold cross validation is performed. Table V presents the results with, in rows 1-4 the first stacking approach and in rows 5-6 the second stacking approach. If one compares the first stacking approach (which consist in learning on the class memberships estimates provided by any standard decoding) with the standard and elimination decodings (Tables III and IV), stacking enables to improve the decoding for all data bases except one (Optdigits). In some cases, the increase of the recognition rate can be spectacular: for PageBlocks with Price or Hastie decoding, replacing $h = argmax$ (Table III) by a decision tree (Table V) results in a doubly of the recognition rate. It highlights the fact that, in a lot of cases, the class membership estimates of some classes are very close one to another and a simple decision rule such as $h = argmax$ cannot cope with such problems: conflicts between classes have to be learned to be overcome. We can also point out that for this first approach to

stacking, the Hastie and the MinMax combination rules seem more appropriate to estimate the class membership estimates. Indeed, among the four stacked combination rules, one of the two above-mentioned always enable to obtain the best results (Table V). If we now consider our second approach to stacking (Table V, rows 5-6) which learns on all the BNN outputs, the performance of stacking depends on the type of predictions provided by the binary classifier (hard or soft predictions). For hard predictions (HP), the prediction of each binary classifier is a 0/1 prediction and for soft predictions (SP), it is the output of each binary classifier. As regards the results of standard decoding (Table IV) and of our first approach to stacking (Table V, rows 5-6), the second approach to stacking using hard predictions (Table V, row 5) does not provide a substantial gain of the recognition rate. On the contrary, using soft predictions (Table V, row 6) enables to obtain better results than standard decoding and than our first approach to stacking on half of the databases. As compared to the work of Savicky [37], the use of posterior probabilities instead of hard class decisions of the binary classifiers to feed the stacking Meta Classifier enables substantial gain in the recognition rate.

To conclude these experiments, using binary classifiers is very interesting since it can be viewed as an ensemble method which performs a simplification of the problem by decomposing it, the latter results been easier to classify by stacking than the initial ones all-at-once or even with standard decoding schemes. The question is now which approach to stacking to choose. Indeed, if stacking always provides better results than standard or elimination decoding, the best stacking approach depends on the data bases. Our second approach to stacking has to deal with very large problems since the size of the feature dimension vector is proportional to the number of classes. For example, for the 26-class Letter dataset, there are $(26 \times 25)/2 = 325$ predictions for each of 16000 examples. This can be a serious drawback. Figure 2 shows the size of the decision tree induced according to the number of classes for our second approach to stacking (dashed lined), one can see that this size rapidly grows with the number of classes and can be very large. This is also the case when using our first approach to stacking whatever the used combination rule (see Figure 2). However, the size of the feature vector is in this case exactly equal to the number of classes. Since no real difference in the size of the induced tree is observed, none of the two approaches to stacking can be preferred, the first being less costly in terms of stacking database size.

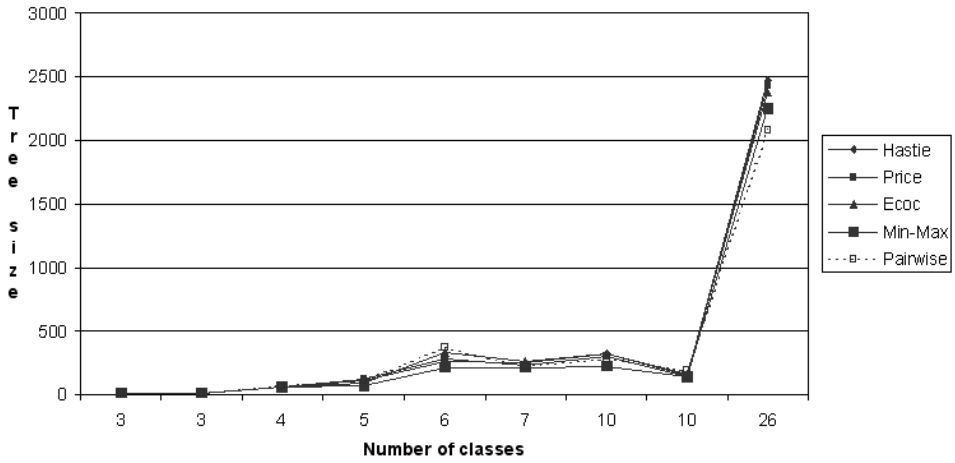

Figure 2. C4.5 Tree size according to the number of classes for the proposed stacking approach: first approach (Hastie, Price, LBD and Min-Max decoding) and second approach (all the pairwise networks outputs).

5. Conclusion

In this paper, we reviewed and evaluated classical methods for multi-class classification based on binary neural networks according to the one-against-one formalism. We have also introduced a new standard decoding method (Min-Max) and a new elimination decoding which are both as suitable as the classical methods presented in the literature as proved by the experiments. The LBD and Min-Max decoding methods have appeared to be the best ones among the tested databases. We show that elimination decoding reduces the error bound of standard decoding methods but standard decoding is more suitable in terms of performance. We also evaluated a technique using stacking decoding where the basic idea is to replace either solely the selection scheme or both the combining rule and selection scheme by a single Meta Classifier that combines therefore either the membership class estimates or all the predictions of the binary classifiers. Using stacking decoding leads to substantial gain in the recognition rate. Future work will concern the use of the set of one-against-one classifiers as a new input sample generator [42] to increase the size of the training dataset of the Meta Classifier when the latter is unbalanced; preliminary results having also shown a new gain in the recognition rate. As regards the results and as it was stated by Furnkranz [12], binary classifiers have now to be considered as an ensemble method such as boosting. Some works have considered this issue in the case of the elimination decoding: since the structure of a DAG is difficult to optimize, it is worth to try to combine

an ensemble of DAG. As stated in [9], since any system of nested dichotomies is biased by imposing a certain order on the set of classes, the class probability estimations will usually differ for two different systems of nested dichotomies. To reduce the error variance, one can build an ensemble of nested dichotomies. A similar work with ensemble of Neural Network Induction Graphs leads to the same results [26]. Pairwise classification has therefore to be considered as an ensemble method and future works will concern this issue: theories from multiple classifier systems have to be used to enhance once again pairwise ensemble classifiers accuracies. Several issues can be considered. First, one can try to combine the outputs of several standard decoding schemes. Second, one can consider the problem as a global optimization one: the tuning of each binary classifier can be performed simultaneously with the optimization of the decoding scheme to cope with the problem of irrelevant classifiers. These research directions place the problem of combining pairwise classifier in the research field of classifier selection and fusion which has not been extensively studied for this particular case.

References

1. Allwein, E., R. Schapire, and Y. Singer: 2000, 'Reducing multiclass to binary: a unifying approach for margin classifiers'. *Journal of Machine Learning Research* **1**(2), 113–141.
2. Bradley, R. A. and M. E. Terry: 1952, 'The rank analysis of incomplete block designs: I. The method of paired comparisons'. *Biometrika* **39**, 324–345.
3. Campbell, C.: 1997, *Constructive Learning Techniques for Designing Neural Network Systems*. San Diego: Academic Press.
4. Cardot, H. and O. Lezoray: 2002, 'Graph of neural networks for pattern recognition'. In: *International Conference on Pattern Recognition (ICPR)*, Vol. 2. pp. 124–127.
5. Crammer, K. and Y. Singer: 2002, 'On the Learnability and Design of Output Codes for Multiclass Problems'. *Machine Learning* **47**(2-3), 201–233.
6. Dietterich, T. and G. Bakiri: 1995, 'Solving Multiclass learning problems via error-coorecting output codes'. *Journal of Artificial Intelligence Research* **2**, 263–286.
7. Ding, C. and I. Dubchak: 2001, 'Multiclass protein fold recognition using support vector machines and neural networks'. *Bioinformatics* **17**, 349–358.
8. Duan, K. and S. Keerthi: 2005, 'Which is the best Multiclass SVM method ? An empirical study'. In: *International Workshop on Multiple Classifier Systems*. pp. 278–285.
9. Frank, E. and S. Kramer: 2004, 'Ensembles of nested dichotomies for multi-class problems'. In: *ICML*. pp. 305–312.
10. Friedman, J.: 1996, 'Another approach to polychotomous classification'. Technical report, Dept. of statistics, Stanford University.

11. Furnkranz, J.: 2002a, 'Pairwise Classification as an Ensemble Technique'. In: *European Conference on Machine Learning (ECML)*. pp. 97–110.
12. Furnkranz, J.: 2002b, 'Round Robin Classification'. *Journal of Machine Learning Research* **2**, 721–747.
13. Hastie, T. and R. Tibshirani: 1998, 'Classification by pairwise coupling'. *The annals of Statistics* **26**(2), 451–471.
14. Hettich, S., C. Blake, and C. Merz: 1998, 'UCI Repository of machine learning databases'. Technical report, University of California, Irvine, Dept. of Information and Computer Sciences.
15. Hsu, C.-W. and C.-J. Lin: 2002, 'A comparison of methods for multi-class support vector machines'. *IEEE Transactions on Neural Networks* **13**(2), 415–425.
16. Ie, E., J. Weston, W. Noble, and C. Leslie: 2005, 'Multi-class protein fold recognition using adaptive codes'. In: *International Conference on Machine Learning (ICML)*. pp. 329–336.
17. Klautau, A., N. Jevtić, and A. Orlitsky: 2002, 'Combined binary classifiers with applications to speech recognition'. In: *International Conference on Spoken Language Processing (ICSLP)*. pp. 2469–2472.
18. Klautau, A., N. Jevtić, and A. Orlitsky: 2003, 'On Nearest neighbor Error-Correcting Output Codes with Application to All-Pairs Multiclass Support Vector Machnies'. *Journal of Machine Learning Research* **4**, 1–15.
19. Knerr, S., L. Personnaz, and G. Dreyfus: 1992, 'Handwritten digit recognition by neural networks with single-layer training'. *IEEE Transactions on Neural Networks* **3**(6), 962–968.
20. Ko, J., E. Kim, and H. Byun: 2004, 'Improved N-division output coding for multiclass learning problems'. In: *International Conference on Pattern Recognition (ICPR)*, Vol. 3. pp. 470– 473.
21. Kressel, U.: 1999, *Advances in Kernel Methods, Support Vector Learning*, Chapt. Pairwise classification and support vector machines. MIT Press.
22. Kwok, T.-Y. and D.-Y. Yeung: 1997, 'Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems'. *IEEE Transactions on Neural Networks* **8**(3), 630–645.
23. Lezoray, O. and H. Cardot: 2001, 'A Neural Network Architecture for Data Classification'. *International Journal of Neural Systems* **11**(1), 33–42.
24. Lezoray, O. and H. Cardot: 2005, 'Combining multiple pairwise neural networks classifiers: a comparative study'. In: *International Workshop on Artificial Neural Networks and Intelligent Information Processing (ANNIIP)*. pp. 52–61.
25. Lezoray, O., A. Elmoataz, and H. Cardot: 2003, 'A color object recognition scheme : application to cellular sorting'. *Machine Vision and Applications* **14**, 166–171.
26. Lezoray, O., D. Fournier, and H. Cardot: 2004, 'Neural network induction graph for pattern recognition'. *Neurocomputing* **C**(57), 257–274.
27. Lu, B.-L. and M. Ito: 1999, 'Task decomposition and Module Combination based on Class Relations: A modular Neural Network for Pattern Classification'. *IEEE Transactions on Neural Networks* **10**(5), 1244–1256.
28. Masulli, F. and G. Valentini: 2000, 'Comparing Decomposition Methods for Classification'. In: *International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, Vol. 2. pp. 788–792.
29. Mayoraz, E. and E. Alpaydin: 1999, 'Support Vector Machines for Multi-class Classification'. In: *International Work conference on Artificial Neural Networks*, Vol. 2. pp. 833–842.

30. Moreira, M. and E. Mayoraz: 1998, 'Improved pairwise coupling classification with correcting classifiers'. In: *European Conference on Machine Learning (ECML)*. pp. 160–171.
31. Ou, G., Y. Murphey, and A. Feldkamp: 2004, 'Multiclass pattern classification using neural networks'. In: *International Conference on Pattern Recognition (ICPR)*, Vol. 4. pp. 585–588.
32. Phetkaew, T., B. Kijsirikul, and W. Rivepiboon: 2003, 'Reordering adaptive directed acyclic graphs: an improved algorithm for multiclass support vector machines'. In: *International Joint Conference on Neural Networks (IJCNN)*, Vol. 2. pp. 1605–1610.
33. Platt, J., N. Cristianini, and J. Shawe-Taylor: 2000, 'Large Margin DAGs for multiclass classification'. In: *Advances in Neural Information Processing Systems (NIPS)*, Vol. 12. pp. 547–553.
34. Price, D., S. Knerr, and L. Personnaz: 1995, 'Pairwise neural network classifiers with propabilistic outputs'. In: *Advances in Neural Information Processing Systems (NIPS)*, Vol. 7. pp. 1109–116.
35. Quinlan, J.: 1993, *C4.5 : programs for machine learning*. San Mateo: Morgann Kauffman.
36. Rifkin, R. and A. Klautau: 2004, 'In defense of one-vs-all classification'. *Journal of Machine Learning Research* **5**, 101–141.
37. Savicky, P. and J. Furnkranz: 2003, 'Combining Pairwise Classifiers with Stacking'. In: *Intellignent Data Analysis (IDA)*. pp. 219–229.
38. Takahashi, F. and S. Abe: 2003, 'Optimizing directed acyclic graph support vector machines'. In: *Artificial Neural Networks in Pattern Recognition (ANNPR)*. pp. 166–173.
39. Tax, D. and R. Duin: 2002, 'Using two-class classifiers for multiclass classification'. In: *International Conference on Pattern Recognition (ICPR)*, Vol. 2. pp. 124–127.
40. Vural, V. and J. G. Dy: 2004, 'A hierarchical method for multi-class support vector machines'. In: *International Conference on Machine Learning (ICML)*. p. 105.
41. Wolpert, D.: 1992, 'Stacked generalization'. *Neural Networks* **5**(2), 241–260.
42. Zhou, Z.-H. and Y. Jiang: 2003, 'NeC4.5: Neural Ensemble Based C4.5'. *IEEE Transactions on Knowledge and Data Engineering* **16**(6), 770–773.