A new artificial neural network ensemble based on feature selection and class recoding

M. P. Sesmero · J. M. Alonso-Weber · G. Gutiérrez · A. Ledezma · A. Sanchis

Abstract Many of the studies related to supervised learning have focused on the resolution of multiclass problems. A standard technique used to resolve these problems is to decompose the original multiclass problem into multiple binary problems. In this paper, we propose a new learning model applicable to multi-class domains in which the examples are described by a large number of features. The proposed model is an Artificial Neural Network ensemble in which the base learners are composed by the union of a binary classifier and a multiclass classifier. To analyze the viability and quality of this system, it will be validated in two real domains: traffic sign recognition and hand-written digit recognition. Experimental results show that our model is at least as accurate as other methods reported in the bibliography but has a considerable advantage respecting size, computational complexity, and running time.

Keywords Classifier ensemble · Multiclass learning · Neural networks · Feature selection · Class recoding

1 Introduction

A supervised learning problem is one in which a given set of examples and their classes, $S = \{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \ldots (\mathbf{X}_n, y_n)\}$, has the objective of finding the f function that allows the prediction of the associated class to new examples: $f(\mathbf{X_i}) = \mathbf{y_i}$. When the set of classes is of finite cardinality and contains more than two elements, the machine learning application receives the name of *multi-class learning* and the f function is called *multi-class* classifier.

The simplest, most cited, and probably most criticized approach used in multi-class classification problems is the so-called *one against all* (OAA) scheme [1 3]. In this model, the k class problem is broken down into k binary classification problems, each of which distinguishes or separates one class from the $(k - 1)$ remaining classes. Therefore, each binary classifier is able to indicate if an example belongs or not to its associated class, but when an example is classified as not belonging to "its class", it is not able to determine to which of the $k - 1$ remaining classes the example belongs to.

This characteristic is implicit to the *one against all* architecture and makes the errors made by only one classifier difficult to rectify. This means that diversity [4] of the binary classifiers (independency in the production of their errors) does not always imply an improvement in the classification given by the ensemble. More detailed analysis is as follows.

Given a *one against all* architecture integrated by k dichotomic classifiers with a binary output ($y_i \in \{0, 1\}$, $i \in \{1, 2, \ldots, k\}$) and suppose that, when trying to classify a certain example, only one of these classifiers catalogues it wrong, that is, the dichotomic classifiers are diverse. Under these circumstances, the possible situations that can arise are

1. Classifier C_j, which makes the error, classifies as negative ($y_j = 0$) an example than should be classified as positive (*false negative*). Since for the rest of the classifiers this example is negative, the output for all of

M. P. Sesmero (✉) · J. M. Alonso Weber · G. Gutiérrez · A. Ledezma · A. Sanchis
Computer Science Department,
Universidad Carlos III de Madrid,
Avda. de la Universidad 30,
28911 Leganés, Madrid, Spain
e mail: msesmero@inf.uc3m.es

them will be {0}, and, therefore, the global output for the ensemble will be {0, …, 0, …, 0}. This means that, the only mechanism to decide the class of the example is a random guess, and therefore the error is not rectifiable.

2. Classifier C_j classifies as positive ($y_j = 1$) an example that should be classified as negative (*false positive*). In this case, two classifiers will exist, C_i and C_j, that will classify the example as belonging to its class ($y_i = 1$ and $y_j = 1$), while the rest of the classifiers will catalogue it as negative ($y_k = 0,\ i \neq k,\ j \neq k$). Consequently, as in the previous situation, it will be impossible to clearly determine to which class the example belongs to.

Therefore, the accuracy of the *one against all* architecture depends principally on the accuracy of the dichotomic classifiers integrating it, but not on its diversity.

To deal with this problem, many schemes based on this architecture use classifiers with output values in the [0, 1] range in order to assign the class label of the classifier with the largest output value [3, 5 7]. Mathematically,

$$c(x) = F(x, y_1, y_2, \ldots, y_k) = \arg\max_{i=1,\ldots,k}(y_i) \quad (1)$$

This modification avoids ties and ambiguous labeling and also creates certain dependence between the diversity of the classifiers and the accuracy of the ensemble. Nevertheless, the correct classification of an example depends heavily on the output value given by a specific classifier (that associated to the class which the example belongs to). That is, $x \in c_i$ will be correctly classified by the ensemble, if and only if, $y_i > y_k\ \forall\ k \neq i$, where y_k is the output value of C_k classifier.

Therefore, relation between the diversity of the classifiers and the accuracy of the ensemble is not always guaranteed, since it depends on a criterion ($y_i > y_k$) that is not faced by the learning algorithms. In this paper, we propose a modification of the *one against all* architecture, in order to rectify this difficulty and to guarantee that the diversity of the base classifiers produces on an increase in the ensembles accuracy.

Another issue that is dealt with in this paper is the need to build a system capable of resolving problems in which the examples are described by a high number of features. In theory, more features should provide more accurate classifiers but, in practice, irrelevant or redundant features may have a negative effect on the accuracy of the classifiers [8 10]. Furthermore, due to running time requirements or constraints imposed by the problem itself, feature selection can be an essential requisite in certain domains. For example, when the designed classification system has to be implemented in hardware as an Artificial Neural Network

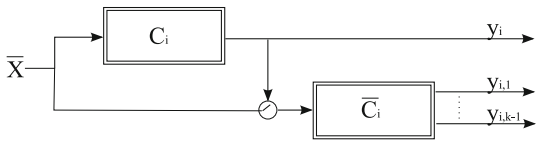

Fig. 1 An illustration of the *i*th base module

on FPGA -*Field Programmable Gate Array*-, a high number of features make the implementation unviable [11].

Following the terminology found in [12], the system proposed here is an ensemble of classifiers in which diversity is obtained manipulating the output targets [3], manipulating the input features [13, 14] and injecting randomness into the learning process [15].

As with the *one against all* architecture, the number of individuals that will integrate the ensemble coincides with the cardinality of the set of classes. The biggest difference compared to the *one against all* architecture is that, in this new model, the ensemble members will not be binary classifiers, but modules composed of a binary classifier and a multi-class classifier (Fig. 1). To simplify terminology, these modules will be called *base modules* and the multi-class classifiers will be called *complementary classifiers*.

Since one of the aims of our work is to develop a system that can be implemented in hardware using FPGA's, prior to the construction of each classifier a feature selection process will be performed. This reduction allows the construction of simpler classifiers and therefore, systems with a reduced learning and classification time. With the objective of increasing models accuracy, the feature subset used to generate the classifiers is not unique but depends on the task associated with each classifier.

The remainder of this paper is organized as follows. In Sect. 2, we present the proposed architecture in detail. Section 3 presents and analyzes the empirical evaluation of this approach. Finally, in Sect. 4, conclusions and future work are summarized.

2 System architecture

The main goal of this paper is to develop a new multiple classifier system characterized by learning from examples belonging to one of k classes ($k \geq 3$) and described by a high number of features. In order to obtain an accurate system, the ensemble will be composed of a number of more simple classifiers in which diversity is obtained:

1. Manipulating the *output* values that are given to the learning algorithm.
2. Varying the feature subsets used to generate the base classifiers.
3. Injecting randomness into the learning process.

In contrast to the multi-class methods based on binary decomposition (*One Against All* (OAA), *One Against One* (OAO) [16, 17], or *Error-Correcting Output Codes* (ECOC) [18]), the ensemble we are proposing is based on the idea that the classifier modules integrating it do not only indicate if an example belongs to one or more specific classes, but they attempt to explicitly indicate the particular class it belongs to. Each module is composed of two classifiers (Fig. 1): the first one, C_i, is a *binary classifier* trained to distinguish if an example belongs to a certain class or not. The second classifier, called *complementary classifier* and named $\overline{C_i}$, is a multi-class classifier with $k-1$ outputs representing classes $\{c_j\}$, where $j \neq i$. This last classifier will only intervene under certain circumstances (see Sect. 2.2).

2.1 Base module construction

In this section, a detailed explanation is given on the design of the base modules and their components: the binary (Sect. 2.1.1) and the complementary classifiers (Sect. 2.1.2). Section 2.2 deals with their interaction.

2.1.1 Binary classifiers

As stated before, the binary classifiers composing the ensemble are analogous to the ones used in the *one against all* architecture. Therefore, each binary classifier is trained with the same data set but with different class labels. To train the *i*th classifier, the training data set Ω_{tr} is decomposed in two sets, $\Omega_{tr} = \Omega_{tr}^{+i} \cup \Omega_{tr}{}^{i}$, where Ω_{tr}^{+i} contains all the class i examples, labeled as "1", and $\Omega_{tr}{}^{i}$ contains all the examples belonging to all other classes, labeled as "0".

One of the premises that must be satisfied by this model is the ability to work with examples described by a high number of features. Practical experience shows that using as much as possible input information (features) does not imply a higher output accuracy. To facilitate and improve the learning process, once the class associated with each example is recoded, the process of determining the subset of most relevant features will take place. This feature selection process has been carried out using the *Weka* tool [19] (version 3.4.12). After analyzing several feature selection methods [20], *Correlation-based Feature Selection* (CFS) [21], with *Best First* [22, 23] as search strategy was chosen as the method for feature selection.

Once the initial data set has been processed and the training set associated with each binary classifier has been generated, the next step is building these classifiers. In our experiments, the classifiers are one hidden layer Neural Networks trained with the Back-Propagation algorithm (Fig. 2) [24], in which:

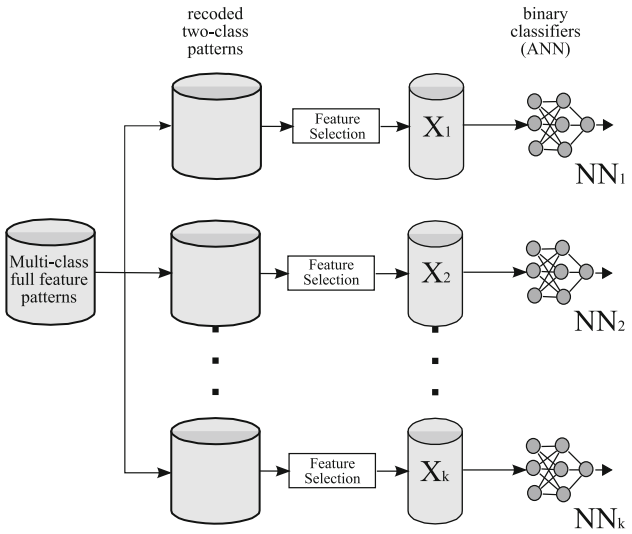

Fig. 2 Construction of the binary classifiers

(a) The number of input neurons depends on the number of features selected by CFS for the corresponding binary problem.
(b) The size of the hidden layer has been experimentally determined by trial and error.
(c) The output layer has a single neuron.

2.1.2 Complementary classifiers

The aim of the complementary classifier, $\overline{C_i}$, is to classify those examples that have been rejected by the corresponding binary classifier (C_i). If the binary classifier is reliable, these examples will belong to one of the $k-1$ classes that the binary classifier learns as negative. Faced with this situation, it seems logical to build $\overline{C_i}$ from examples that belong to these $k-1$ classes. However, if the binary classifier produces a false negative, the complementary classifier will be forced to classify an example that does not belong to any of the classes it has learned. To guarantee that $\overline{C_i}$ does not generate outputs that might produce ambiguity or conflictive situations, these classifiers will be trained with examples belonging to all classes.

To achieve diverse classifiers with $k-1$ outputs, before the construction of $\overline{C_i}$ classifier, a new recoding of the class associated with each example will be performed. Specifically, the coding associated with the examples belonging to the *i*th class will be a vector of $k-1$ components where all the values will be "0". On the other hand, the codification associated with the examples belonging to the *j*th class will be carried out establishing a correspondence between the different classes and the $k-1$ components that make up the output vector. Therefore, for a domain in which there are three classes, c_1, c_2, c_3, the codification associated with the learning examples $\mathbf{X_i}$ linked to the $\overline{C_1}$ classifier will be:

$\{0,0\}$ if $\mathbf{X_i} \in c_1$
$\{1,0\}$ if $\mathbf{X_i} \in c_2$
$\{0,1\}$ if $\mathbf{X_i} \in c_3$

With the objective of providing these classifiers with a larger diversity, the feature selection process has been performed considering only the examples associated with $k-1$ classes. Proceeding in this way, complementary classifier will be built on different training sets, and consequently will be more diverse.

As with the binary classifiers, the complementary classifiers will be implemented as one hidden layer neural networks. In this case, the number of output nodes will be $k-1$ (Fig. 3).

2.2 Classifier integration

Once the classifiers that make up a base module have been built, the next step is to determine their interrelation. In order to do so, three options have been analyzed:

(a) *Parallel combination* When this architecture is applied, the output given by the classifying module associated with the *i*th class is a vector of *k* components ($Y_i(x) = [y_1, y_2, \ldots, y_k]$) in which the y_i component is generated by the binary classifier and the rest of components by the complementary classifier. Therefore, to find out the class associated with an example, it is necessary to know the output generated by both the binary classifier and the complementary classifier.

(b) *Serial combination* The complementary classifier only intervenes when the binary classifier (C_i) classifies the example as not belonging to "its class". In this case, the output given by the base module linked to the *i*-th class will be the same to that described for the parallel combination. Otherwise, the output of this module will be a vector in which the only component different from zero will be that generated by C_i ($Y_i(x) = [0, 0, \ldots, y_i, \ldots, 0]$).
Since the output of the binary classifier is a continuous value within the [0,1] range, the critical point in this scheme is determining the threshold (θ) that (1) allows a binary classifier to discern if an example belongs or not to its associated class ($\boldsymbol{x} \in c_i$ if $y_i(\boldsymbol{x}) > \theta$) and, (2) minimizes the ensemble error.
Figure 4 shows, for the domains analyzed in Sect. 3, the relationship between ensemble accuracy and the threshold value. The output of the ensemble is computed averaging the outputs given by each base module and choosing then the one of highest value. Mathematically,

$$C(x) = \arg\max_{i=1}^{k} \left(\frac{\sum_{j=1}^{k} y_{ji}}{k} \right) \tag{2}$$

where y_{ji} is the *j*th output of the *i*th module and *k* the number of classes.
There are several things to note. First, the serial combination is equivalent to the OAA architecture for a threshold value of $\theta = 0$, and is equivalent to the parallel architecture when the threshold is $\theta = 1$. Second, serial and parallel combinations perform equally well for all threshold values $\theta \geq 0.05$. Indeed, the statistical tests used in Sect. 3 cannot distinguish them.

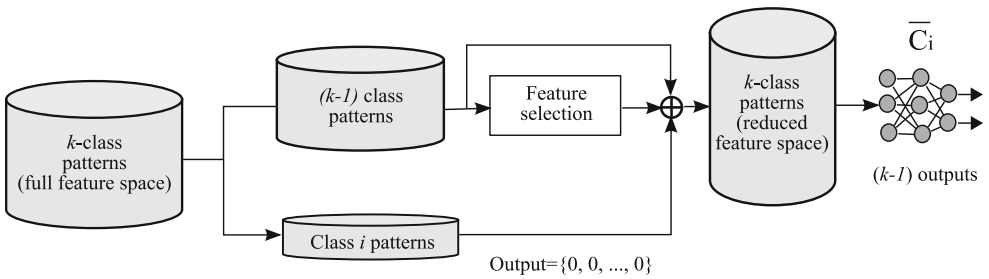

Fig. 3 Structure of the *i*th complementary classifier ($\overline{C_i}$)

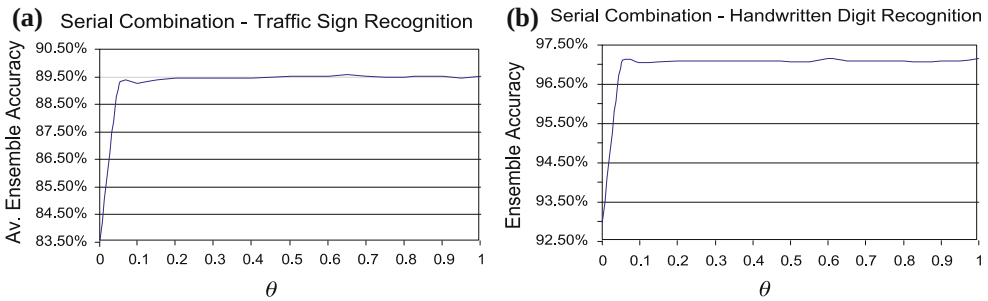

Fig. 4 Relationship between ensemble accuracy and θ for: **a** Traffic sign recognition and **b** Handwritten digit recognition. For θ 0 the architecture is equivalent to an *OAA architecture* and for θ 1 it corresponds to the *parallel combination*

(c) *Hierarchical combination* The intervention of the complementary classifiers depends on the result given by the classifiers that compose the OAA *architecture*. In other words, it establishes a hierarchical dependence between the ensemble made up of binary classifiers (*OAA architecture*) and the performance of the complementary classifiers: If the *OAA* architecture is capable of unequivocally classifying an example, the complementary classifiers will not be used. On the other hand, examples ambiguously labeled by the OAA architecture will be sent to the complementary classifiers.

From this perspective, the *OAA* architecture is considered capable of unequivocally classifying an example when the y_i ($i \in \{1, 2, \ldots, k\}$) binary classifier outputs satisfy the following relation:

$$\exists i / y_i > \theta_1 \quad \text{and} \quad y_j < \theta_2 \, \forall j \neq i \tag{3}$$

where θ_1 and θ_2 are different thresholds/$\theta_1 \geq \theta_2$.

Figure 5 shows that the best performance region is defined for the threshold values $\theta_1 = 1$ or $\theta_2 = 0$. For both cases, the hierarchical architecture is equivalent to the parallel combination. Figure 6 illustrates the binary-complementary integration schemes analyzed in this work.

3 Empirical evaluation

3.1 Data and methods

The proposed system has been tested in two real domains: *Traffic sign recognition* and *Hand-written digit recognition*.

In both domains, the patterns are images in PGM (*Portable Gray Map*) format. The number of features (pixels) used to describe each instance (image) is: 1,024 (32 × 32) for traffic signs and 784 (28 × 28) for hand-

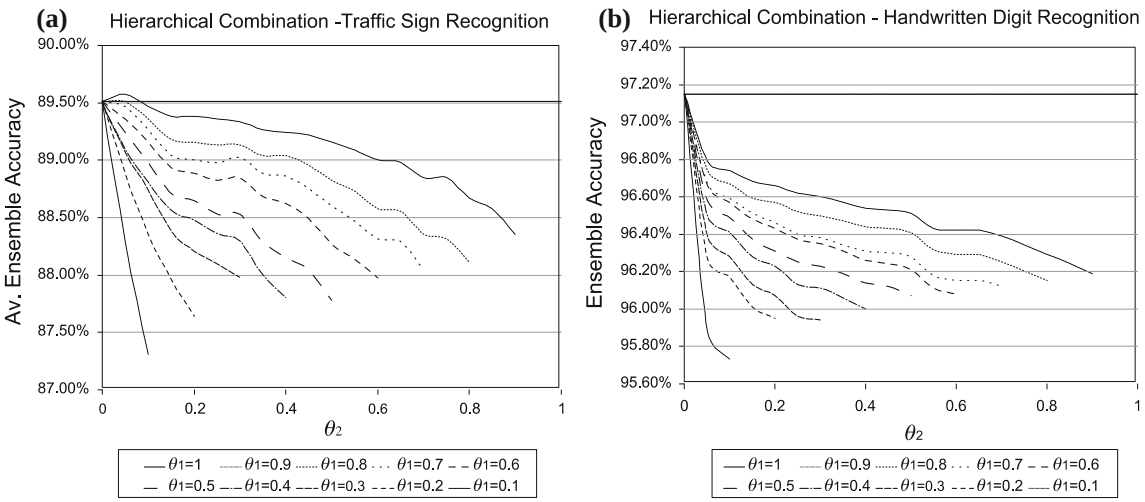

Fig. 5 Relation between θ_1 and θ_2 and the ensembles accuracy for **a** Traffic sign recognition and **b** Handwritten digit recognition. θ_1 1 and θ_2 0 equivalent to *parallel combination*

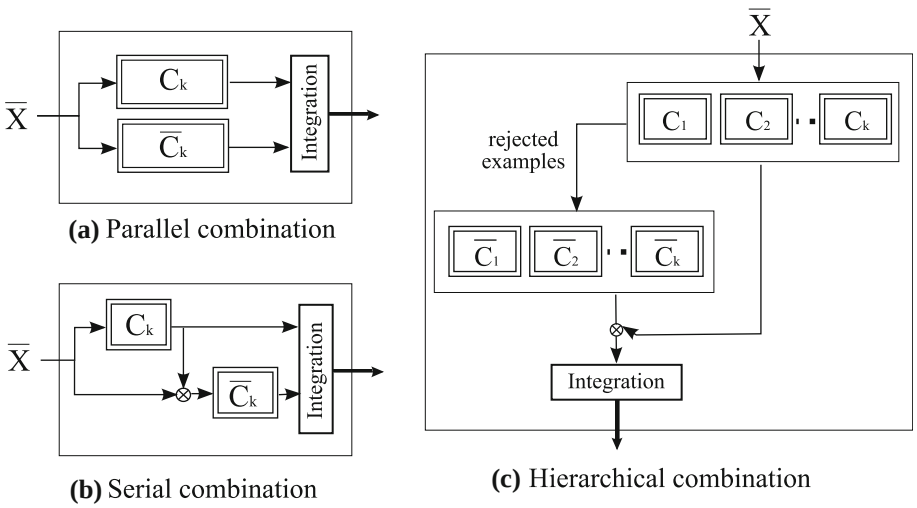

Fig. 6 Binary complementary integration: **a** Parallel; **b** Serial; **c** Hierarchical

Table 1 Summary of the 6 measures of diversity used

Name	Symbol	Definition	P	↑/↓
Plain disagreement measure	Plain	$\frac{1}{N}\sum_{n}^{N}{}_{1}\,\mathrm{Diff}(C_i(x_n), C_j(x_n))$	Y	↑
Fail/non fail disagreement measure	Dis	$\frac{N^{01}+N^{10}}{N^{11}+N^{10}+N^{01}+N^{00}}$	Y	↑
Q statistic	Q	$\frac{N^{11}N^{00}-N^{01}N^{10}}{N^{11}N^{00}+N^{01}N^{10}}$	Y	↓
Correlation coefficient	ρ	$\frac{N^{11}N^{00}-N^{01}N^{10}}{\sqrt{(N^{11}+N^{10})(N^{01}+N^{00})(N^{11}+N^{01})(N^{10}+N^{00})}}$	Y	↓
Kappa degree of agreement statistic	κ	$\frac{\sum_{i=1}^{k}\frac{N_{ii}}{N}-\sum_{i=1}^{k}\left(\frac{N_{is}N_{si}}{N\ N}\right)}{1-\sum_{i=1}^{k}\left(\frac{N_{is}N_{si}}{N\ N}\right)}$	Y	↓
Ambiguity	Amb	$\frac{1}{LNK}\sum_{l}^{L}{}_{1}\sum_{n}^{N}{}_{1}\sum_{k}^{K}{}_{1}\left(\mathrm{Is}\,(C_l(x_k)\quad k)\quad \frac{N_k^n}{L}\right)^2$	N	↑

The arrow specifies whether diversity is greater if the measure is lower (↓) or greater (↑). 'P' stands for 'Pairwise' measures

N is the number of instances in the data set; L is the number of base classifiers; K is the number of classes; N^{ab} is the number of instances in the data set, classified correctly (a 1) or incorrectly (a 0) by the classifier i, and correctly (b 1) or incorrectly (b 0) by the classifier j; N_{ij} is the number of instances in the data set, recognized as class i by the first classifier and as class j by the second one; $C_i(x_n)$ is the class assigned by classifier i to instance x_n N_k^n is the number of base classifiers that assign instance n to class k; $Is()$ is a truth predicate

written digits. In order to analyze the viability and the quality of the proposed system, its results will be compared to those obtained by:

(a) A single neural network with one hidden layer and k output nodes (k = number of classes in each domain).
(b) A system of k Neural Networks modeled using an *OAA scheme*.
(c) Bagging [25] with ANN as base classifiers. As a means of determining the number of base classifiers in this architecture, an attempt has been made to reach a balance between Quinlan's 10 replicas [26], Breiman's 50 replicas [25] and the computational cost of training a neural network. Based on this data, and after an analysis of the relationship between the number of classifiers and the error rate of the system, carried out during some of the experiments, we finally chose 20 as the best number of replicas. This is quite close to the number of replicas proposed by Opitz and Maclin [27] who assert that whenever Bagging is implemented with Neural Networks, the largest error-reduction rate occurs when using between 10 and 15 base classifiers. In order to make result comparable [28], the average Eq. (2) will be used as combining rule.

Therefore, both domains will be evaluated comparing the proposed new model (using the parallel combination) with three classification models: single NN, *OAA architecture*, and *Bagging*, all with feature selection. Additionally, some results with these three models without feature selection are included as a reference.

The feature selection processes have been carried out considering solely the training examples and have been performed using the Weka tool. Once the most relevant feature subset is determined, it will be regarded as a specific idiosyncratic parameter of the corresponding classifier. In order to achieve this objective among others, the neural networks used in each model have been implemented using a software simulator written in C++ and developed by the authors of this paper.

In order to analyze the influence of the diversity of the base classifiers on the ensemble accuracy, some well-known measures of diversity [29, 30] will be computed. Table 1 shows a summary of the used measures of diversity, their types (pairwise or non-pairwise), and the theoretic relationship between diversity and accuracy of the ensemble.

For all pairwise measures (plain disagreement, fail/non-fail disagreement, the Q statistic, the correlation coefficient and the kappa statistic), ensemble diversity is equal to the averaged value over all pairs of classifiers:

$$M_{av} = \frac{2}{L(L-1)}\sum_{i=1}^{L}{}^{1}\sum_{k=i+1}^{L} M_{i,k} \tag{4}$$

In the following subsections, there is a description of the evaluated domains, as well as a detailed explanation of the characteristics of the distinct systems implemented and the experimental results obtained in each of them.

3.2 Traffic sign recognition

The reason for our interest in this domain is the need to build Advanced Driver Assistance Systems (ADAS), which have, among other functions, the capacity to warn drivers of potential dangers, avoid or induce the realization of certain manoeuvres and limit driving speed. The need for integration of the classification module into a ADSS capable of operating in real time poses certain design

Fig. 7 Examples for the nine different traffic sign classes

restrictions which make its construction difficult. For example, in order to embed the designed software system as a FPGA hardware implementation, the classification algorithm must be based on neural networks with a limit of 200 nodes for each network.

This domain contains 900 images of prohibition traffic signs distributed evenly among nine classes (Fig. 7): no pedestrian entry, no left/right turn, no stopping or parking, no passing and the signs limiting the speed to 20 40 50 60 and 100 km/h.

To create the data set, each image has been scaled to 32×32 pixels and saved in PGM format. This preprocessing allows us to describe each pattern with a 1,033-component vector, in which the first 1,024 elements represent the grayscale values for each pixel, and the last nine elements codify the corresponding sign class.

To evaluate the accuracy of the implemented models and determine if the differences among them are statistically significant, we have used a statistical test called *combined 5 × 2cv F-test* [31]. This test is based on performing five replications of twofold cross-validation. In each replication, the available data are randomly partitioned into two equal sized sets. Then, each learning algorithm is trained on one data set and tested on the other. If $p_i^{(j)}$ is the difference between the error rates of the two classifiers on fold j of replication i and $s_i^2 = \left(p_i^{(1)} - p_i\right)^2 + \left(p_i^{(2)} - p_i\right)^2$ is the estimated variance on replication i ($p_i = \left(p_i^{(1)} + p_i^{(2)}\right)/2$), then, the statistic:

$$F \text{ - test} = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}\left(p_i^{(j)}\right)^2}{2\sum_{i=1}^{5} s_i^2} \tag{5}$$

is approximately F distributed with 10 and 5 degrees of freedom. Therefore, we reject the hypothesis that the two classifiers have the same error rate with 0.95 confidence if *F-test* is greater than 4.735.

Once the data sets that will be used in the experimental part have been determined, the next step is to determine the topology of classifiers that integrate each model. After running several different experimental tests [32], we decided to use neural networks in which the size of the hidden layer depends on the amount of neurons present in the input layer, that is, on the cardinality of the feature space. Thus, the number of hidden neurons varies from 50 for a full feature space, to 30 for a downsized input space through feature selection.

Table 2 shows the performance obtained by the proposed new model compared with a single neural network, an OAA architecture and Bagging when they are built using:

(a) The full feature space 1,024 features.
(b) The feature subsets obtained when applying *CFS*. For both, Bagging and the neuronal network with nine output nodes, the cardinality of these subsets is 129 ± 10. The number of features selected in the construction of the binary and the complementary classifiers associated with the different classes is shown in Table 3.

If we apply the F-test (Table 5) on the results derived from executing 5×2cv on each of the classification models (Table 4), it can be deduced at a 0.95 significance level that the proposed model is:

1. Statistically better than any of the other models with feature selection (F-test is greater than 4.735 and the new models accuracy is the highest).
2. Statistically equivalent to all those models which are built using the full feature space (F-test <4.735).

For those systems with the best performance, Table 6 summarizes the ensembles classification time (on a 2.5 GHz Intel Xeon) and other main characteristics of the base learners (training time, number and size). According to these values, we can conclude that compared with traditional OAA and Bagging method, this new model improves in size and running time. Compared with the single full feature space ANN, the running time is slightly higher, but it should be considered that the new model is simulated serially. Considering its highly parallelizable nature, the running time can be reduced by a factor equivalent to the number of individual base modules.

In order to analyze the influence of diversity of the base classifiers on the ensembles accuracy, several measures of diversity are computed for *Bagging* (with and without feature selection) and for the new proposed model (results are shown in Table 7). It is interesting to note that measures where low values indicate high diversity (Q, ρ and κ) have high values, whereas measures where high values indicate high diversity (*plain*, *dis* and *amb*) have low values. As reported in [33], this could be an indication of the lack of any strong relationship between diversity measures and ensemble accuracy in real-life classification problems.

Table 2 Performance of the evaluated models on the traffic sign classification problem

	Original feature space			Feature subspace selected by CFS			
	1 ANN (%)	OAA (%)	Bagging (%)	1 ANN (%)	OAA (%)	Bagging (%)	New model (%)
C1	98.40	**98.60**	**98.60**	98.20	97.60	98.20	98.20
C2	98.60	**99.80**	99.40	98.20	98.80	98.60	99.40
C3	91.20	90.20	**95.80**	83.00	80.00	85.80	89.20
C4	93.20	93.20	93.80	93.20	87.80	93.00	**94.80**
C5	**95.60**	94.80	95.20	94.00	91.40	94.60	94.80
C6	75.80	**77.40**	75.80	67.60	58.60	70.00	75.40
C7	81.60	**82.40**	**82.40**	81.40	80.40	79.60	**82.40**
C8	87.80	89.40	88.80	88.20	87.20	**91.00**	90.80
C9	87.60	86.20	**87.80**	77.20	69.80	77.40	80.60
Total	89.98	90.22	**90.84**	86.78	83.51	87.58	89.51

The values for *Ci* indicate the true positive rate (recall) for class *Ci* examples. The values for Total indicate the models accuracy. The models on the left have been built using the full feature space and the models on the right have been built using feature selection. Bold face type indicates the best values in each category

Table 3 Number of selected features in the construction of the binary and the complementary classifiers

Class	Road sign type	Number of selected features	
		Binary classifier	Complementary classifier
C1	No pedestrians	72 ± 8	119 ± 7
C2	No (left, right) turn ahead	71 ± 11	144 ± 9
C3	No stopping and no parking	67 ± 13	115 ± 9
C4	No passing	73 ± 11	118 ± 12
C5	60 km speed limit	91 ± 8	152 ± 12
C6	50 km speed limit	65 ± 12	126 ± 13
C7	40 km speed limit	81 ± 15	130 ± 9
C8	20 km speed limit	75 ± 15	140 ± 11
C9	100 km speed limit	65 ± 11	129 ± 8

In the first column appears the label of each class

Table 4 Errors committed by each model shown fold wise (450 examples per fold)

Fold	1 ANN 1024	1 ANN CFS	OAA 1024	OAA CFS	Bagging 1024	Bagging CFS	New model
1 1	46	70	42	84	37	55	46
1 2	42	63	42	74	29	53	50
2 1	45	50	39	66	37	50	39
2 2	52	64	44	85	45	53	44
3 1	50	62	49	80	49	59	49
3 2	40	56	39	67	42	54	51
4 1	51	59	48	68	44	67	57
4 2	46	52	48	75	42	57	39
5 1	44	59	47	77	49	58	49
5 2	49	61	42	66	38	53	48
Mean	46.5	59.6	44	74.2	41.2	55.9	47.2

Table 5 Performance comparison between two models M1 (horizontally) and M2 (vertically) through the variation of the combined 5 × 2cv *F* test

	1 ANN 1024	1 ANN CFS	OAA 1024	OAA CFS	Bagging 1024	Bagging CFS	New model
1ANN 1024	X	23.62 (+)	1.30 (.)	13.81 (+)	1.99 (.)	6.44 ( )	0.95 (.)
1ANN CFS		X	11.66 ( )	5.40 (+)	17.65 ( )	2.99 (.)	4.80 ( )
OAA 1024			X	24.48 (+)	2.30 (.)	11.29 ( )	0.85 (.)
OAA CFS				X	47.50 ( )	5.68 ( )	6.09 ( )
Bagging 1024					X	14.77 (+)	1.50 (.)
Bagging CFS						X	5.51 ( )
New model							X

A (.) symbol indicates that M1 and M2 are statistically equivalent (F test ≤ 4.71). A (+) symbol indicates that the M1 model significantly outperforms model M2. A ( ) symbol indicates the contrary

Table 6 Ensembles classification time on a 2.5 GHz intel Xeon and details (training time, number, and size) of base learners. In the New model the number of neurons and weights for both the binary and the complementary classifiers are shown

	Classification time (450 instances)	Training time (500 cycles)	Number of base learners	Input units	Weights (layer 1)	Hidden units	Weights (layer 2)	Output units
1 ANN (1024)	0m0.244 s	1m32.178 s	1	1,024	51,200	50	450	9
Bagging (1024)	0m1.288 s	1m32.178 s	20	1,024	51,200	50	450	9
OAA (1024)	0m0.684 s	1m31.818 s	9	1,024	51,200	50	50	1
New model	0m0.292 s	0m12.757 s	9	74 ± 13	2,226 ± 408	30	30	1
			9	131 ± 15	3,921 ± 463	30	240	8

3.3 Hand-written digit recognition

The MNIST data collection (available at http://yann.lecun.com/exdb/mnist/) was used to assess the system's capacity to recognize hand-written digits.

This data collection [34] has 60,000 training examples and 10,000 test examples. Hand-written digits are represented as grayscale 28 × 28-pixel images and belong to 10 different classes (Fig. 8).

As in the previous case, the proposed system will be compared to the following classification models:

(a) A single classifier
(b) An OAA architecture
(c) Bagging formed by 20 base classifiers.

All these models were implemented by using one hidden layer neural networks with Back-propagation as learning algorithm. On the other hand, the number of neurons present in the input layer depends on whether CFS is applied or not. Whenever no feature selection process is applied, the number of input nodes is 784. On the other hand, if Bagging or the single NN are combined with CFS the number of inputs is 218. Finally, the number of neurons in the binary classifiers and in their corresponding complementary classifiers when CFS is applied is shown on Table 8.

Due to the large amount of examples and input features, the learning phase in this domain is very time-demanding. Therefore, the number of neurons present in the hidden layer has been previously established according to the number of input and output neurons. Thus, when CFS is applied, binary classifiers have 15 hidden neurons, whereas multiclass classifiers have 50. On the other hand, when working with the full feature space (784 features), binary classifiers have 50 hidden neurons, whereas multiclass classifiers have 100.

In this domain, the statistical comparison of the different classification models will be done using the *McNemar Test*. This test is considered to be less reliable than the *F*-test, but it is also regarded as the best alternative whenever the computational cost of the experiment does not allow a cross-validation process.

The *McNemar Test* [35], as well as the *F*-test, enables a comparison of two classifiers, f_A and f_B, when they are trained and tested on the same data sets. It has the following formulation: If

n_{00} is the number of examples misclassified by both f_A and f_B
n_{01} is the number of examples misclassified by f_A but not by f_B
n_{10} is the number of examples misclassified by f_B but not by f_A
n_{11} is the number of examples misclassified by neither f_A nor f_B

Table 7 Traffic sign recognition. Diversity values for each ensemble shown fold wise (450 examples per fold)

Fold	Model	Plain	Dis	Q	ρ	κ	Amb
1 1	New model	0.177	**0.136**	0.883	0.503	0.800	0.018
	Bagging 1024	**0.178**	0.135	**0.880**	**0.498**	**0.799**	**0.019**
	Bagging BFCFS	0.162	0.129	0.902	0.550	0.817	0.017
1 2	New model	**0.158**	0.121	0.896	0.506	**0.822**	**0.016**
	Bagging 1024	0.156	**0.123**	**0.879**	**0.474**	0.824	**0.016**
	Bagging BFCFS	0.137	0.096	0.946	0.634	0.846	0.014
2 1	New model	**0.170**	**0.119**	0.902	0.525	**0.808**	**0.017**
	Bagging 1024	0.162	0.116	**0.901**	**0.519**	0.817	**0.017**
	Bagging BFCFS	0.138	0.098	0.937	0.598	0.844	0.015
2 2	New model	**0.191**	**0.133**	0.899	0.542	**0.784**	**0.019**
	Bagging 1024	0.173	0.131	**0.882**	**0.502**	0.805	0.018
	Bagging BFCFS	0.144	0.109	0.930	0.594	0.838	0.015
3 1	New model	0.157	0.106	0.933	0.601	0.823	0.015
	Bagging 1024	**0.169**	**0.122**	**0.907**	**0.549**	**0.810**	**0.018**
	Bagging BFCFS	0.143	0.100	0.947	0.646	0.838	0.015
3 2	New model	**0.180**	**0.130**	**0.881**	**0.500**	**0.798**	**0.018**
	Bagging 1024	0.174	0.125	0.894	0.515	0.804	**0.018**
	Bagging BFCFS	0.138	0.103	0.938	0.611	0.844	0.015
4 1	New model	0.164	0.116	0.916	0.557	0.815	0.016
	Bagging 1024	**0.176**	**0.131**	**0.888**	**0.510**	**0.802**	**0.019**
	Bagging BFCFS	0.146	0.106	0.940	0.627	0.835	0.015
4 2	New model	0.166	0.122	0.905	0.539	0.813	0.016
	Bagging 1024	**0.172**	**0.133**	**0.872**	**0.484**	**0.806**	**0.018**
	Bagging BFCFS	0.146	0.106	0.933	0.598	0.835	0.015
5 1	New model	**0.175**	**0.129**	**0.894**	0.525	**0.802**	0.017
	Bagging 1024	0.174	**0.129**	**0.894**	**0.524**	0.804	**0.018**
	Bagging BFCFS	0.156	0.119	0.918	0.574	0.824	0.016
5 2	New model	**0.171**	**0.121**	0.903	0.534	**0.807**	**0.017**
	Bagging 1024	0.157	0.118	**0.894**	**0.503**	0.823	**0.017**
	Bagging BFCFS	0.136	0.098	0.942	0.622	0.847	0.014

Low values of Q, ρ and κ *correspond to high diversity. High values of plain*, *dis* and *amb correspond to high diversity*. Bold face type indicates the algorithm with the highest diversity

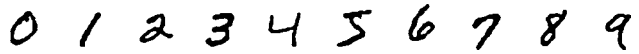

Fig. 8 Examples from the MNIST database

then:

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \tag{6}$$

is a *Chi-square* statistic with one degree of freedom. Therefore, if $\chi^2 \leq 3.84$ is satisfied, then we can admit, with a 0.05 significance level, that the classifiers have the same percentage error.

Tables 9, 10, 11, 12, 13 sum up the experimental results obtained when implementing the aforementioned models.

The conclusion of the analysis of the values shown in Tables 9 and 10 is that the performance of the proposed new model is equivalent to those of *Bagging* and of the *OAA* architecture when they work with the whole set of features. As the decrease in the number of features defining the examples entails a drastic reduction of the testing and training time (Table 13), we can conclude that the proposed system is as precise as *Bagging* and the *OAA* architecture, but clearly much more efficient. Due to the lower performance of the single full feature space ANN, the evaluation of its running time is not appropriate.

According to the measures of diversity used in this study (Table 11), the base classifiers that compose both *Bagging* and the proposed model are not very diverse. Nevertheless, Table 12 shows an accuracy improvement of the ensemble in relation to the base classifiers.

4 Conclusions

In this paper, we have proposed a new classification architecture that efficiently resolves problems where the

Table 8 Number of selected features

Digit	Number of selected features	
	Binary classifier	Complementary classifier
0	70	192
1	118	206
2	97	210
3	104	203
4	77	210
5	47	208
6	73	185
7	71	200
8	84	203
9	87	219

In the first column appears the label of each class

examples belong to one of k classes ($k > 2$) and are described by a high number of features. This architecture can be considered a classifier ensemble in which each component is also an ensemble. The latter ensembles have been called base modules and are composed of a binary classifier and a multi-class classifier which we have designated as complementary classifier. Following Dietterich's terminology, the diversity of the base modules is obtained modifying the class label associated to each example, varying the feature subspace used to generate the ensemble members and injecting randomness into the learning process (random weight setting in the *Backpropagation* algorithm for training neural networks).

Table 11 Diversity values for bagging (with and without feature selection) and for the new model

Model	Plain	Dis	Q	ρ	κ	Amb
New model	0.059	**0.049**	**0.952**	**0.491**	0.934	0.005
Bagging 1024	0.045	0.037	0.968	0.531	0.950	0.004
Bagging BFCFS	**0.060**	**0.049**	0.957	0.523	**0.933**	**0.006**

Bold face type indicates the algorithm with the highest diversity

Table 9 Accuracy on the test set (%) for the evaluated classification methods

	Original feature space			Feature subspace selected by CFS			
	1ANN 784 × 100 × 10	OAA 784 × 50 × 1	Bagging 784 × 100 × 10	1ANN 218 × 50 × 10	OAA CFS × 15 × 1	Bagging 218 × 50 × 10	New model
C0	98.06	98.78	99.08	98.47	98.06	98.98	99.08
C1	98.85	99.12	98.85	98.50	98.50	98.68	98.77
C2	95.54	97.38	96.90	92.83	91.67	95.74	96.32
C3	95.05	97.33	97.82	93.86	92.57	95.84	97.33
C4	96.84	97.25	97.56	94.81	94.20	96.54	97.15
C5	95.18	95.96	96.52	91.82	86.88	94.73	96.08
C6	96.97	97.91	97.49	95.51	94.05	97.18	98.02
C7	95.91	96.50	96.50	94.16	92.22	95.62	96.79
C8	95.59	96.30	96.71	93.02	88.19	95.89	97.02
C9	95.24	95.44	95.44	91.58	92.07	94.85	94.75
Total	96.36	97.23	97.31	94.52	92.97	96.44	97.15

Table 10 Errors and values for the McNemar test

	1ANN 784 × 100 × 10	1ANN 218 × 50 × 10	OAA 784 × 50 × 1	OAA CFS × 15 × 1	Bagging 784 × 100 × 10	Bagging 218 × 50 × 10	New model
1ANN 784 × 100 × 10	**364**	220	172	228	202	209	187
1ANN 218 × 50 × 10	70.95 ( )	**548**	192	287	208	272	226
OAA 784 × 50 × 1	24.90 (+)	165.31 (+)	**277**	192	175	191	174
OAA CFS × 15 × 1	186.98 ( )	35.03 ( )	303.06 ( )	**703**	211	250	225
Bagging 784 × 100 × 10	38.59 (+)	192.73 (+)	0.25 (.)	340.89 (+)	**269**	221	205
Bagging 218 × 50 × 10	0.16 (.)	101.34 (+)	24.24 ( )	214.16 (+)	40.42 ( )	**356**	250
New model	22.12 (+)	180.17 (+)	0.23 (.)	323.21 (+)	1.56 (.)	34.75 (+)	**285**

The values of the diagonal (shown in bold) correspond to the errors committed by each model. Values above the diagonal show the common errors for two experiments (n_{00}). Values under the diagonal represent the values obtained with Eq. (6). A (+) symbol indicates that model M1 (horizontally) significantly outperforms model M2 (vertically), a ( ) symbol indicates the contrary. A (.) symbol stands for equivalent models

Table 12 Ensemble accuracy and the average and maximum base classifier accuracy for *bagging* (with and without feature selection) and the new model

Model	Ens. Acc (%)	Max Acc (%)	Ave. Acc (%)	Ens. Acc Max Acc (%)	Ens. Acc Ave Acc (%)
New model	97.15	95.50	94.96	**1.65**	**2.19**
Bagging 1024	97.31	96.17	95.84	1.14	1.47
Bagging BFCFS	96.44	95.02	94.51	1.42	1.93

Columns 4 and 5 show the differences between the ensemble accuracy and the average/maximum base classifier accuracy. Best values are shown in bold face

Table 13 Classification time of the ensemble and details (training time, number and size) of base learners. In the New model the number of neurons and weights for both the binary and the complementary classifiers are shown

	Test time (1,000 instances)	Training time (500 cycles)	Num of base learners	Input units	Weights (layer 1)	Hidden units	Weights (layer 2)	Output units
1 ANN (784)	0 m.03.667	262m39.704 s	1	784	78,400	100	1,000	10
Bagging (784)	0m29.206 s	262m39.704 s	20	784	78,400	100	1,000	10
OAA (784)	0m08.257 s	165m7.717 s	10	784	78,400	100	100	1
New model	0m05.360 s	42m33.332 s	10	82 ± 20	1,242 ± 300	15	15	1
			10	204 ± 10	10,180 ± 482	50	450	9

Furthermore, it can be concluded that, unlike other models, the proposed architecture uses three of the five techniques (Bayesian Voting, Manipulating the Training Examples, Manipulating the Input Features, Manipulating the Output Targets and Injecting Randomness) suggested by Dietterich to guarantee the diversity of the ensemble.

In this new architecture, the feature selection process is carried out using a supervised method, which is independent of the learning algorithm (filtering method) used in the classifier construction**.** In other words, the feature set chosen in each case depends on what examples are used in training and on the label class associated with each of them, but not on the learning algorithm inherent to each classifier.

Another advantage of this system compared to other models like *Bagging*, is that the number of classifiers integrating the ensemble is predetermined. That is, the proposed model avoids one of the main drawbacks that the design of a large number of classifier ensembles presents: determining the number of base classifiers optimizing the set [36 39]. On the other hand, and contrary to other models like Boosting [40], the independence of the classifiers that make up the ensemble allows the classifiers to be constructed simultaneously on distributed computers, shortening the total training time.

The experimental results indicate that our model's accuracy is comparable to the one obtained with other classic classification methods (a single Neural Network, *Bagging* and *OAA*). However, the drastic decrease in the number of features that describe the examples improves our model with regard to size, computational complexity, and running time. These improvements make the presented model not only a good proposal from the software point of view, but it can also be considered a good alternative to the construction of real time working systems implementing the classifiers with FPGA's [11].

In the future, we intend to evaluate the quality of the system in other domains and analyze the dependence between the proposed model and the algorithm used in the construction of the classifiers integrating the ensemble.

Acknowledgments The research reported here has been supported by the Spanish MCyT under project TRA 2007 67374 C02 02.

References

1. Rangachari A, Mehrotra K, Chilukuri KM, Rank S (1995) Effi cient classification for multiclass problems using modular neural networks. Trans Neural Netw 6(1):117 124
2. Rifkin R, Klautau A (2004) In defense of one vs all classification. J Mach Learn Res 5:101 141
3. Ou G, Murphey L (2007) Multi class pattern classification using neural networks. Pattern Recognit 40(1):4 18
4. Hansen L, Salamon P (1990) Neural network ensembles. IEEE Trans Pattern Anal Mach Intell 12(19):993 1001
5. Tax DMJ, Duin RPW (2002) Using two class classifiers for multiclass classification. 16th International Conference on Pattern Recognition, vol 2
6. Hard Peled S, Roth D, Zimak D (2003) Constraint classification for multiclass classification and ranking. In: Proceedings of the 16th Annual Conference in Neural Information Processing Sys tems (NIPS), pp 785 192

7. García Pedrajas N, Ortiz Boyer D (2003) Improving multiclass pattern recognition by the combination of two strategies. Trans Pattern Anal Mach Intell 28(6):1001 1006
8. Liu H, Yu L (2002) Feature selection for data mining, research technical report. Available in: http://www.public.asu.edu/∼huanliu/sur fs02.ps
9. Oliveira LS, Sabourin R, Bortolozzi F, Suen CY (2003) A methodology for feature selection using multi objective genetic algorithms for handwritten digit string recognition. Int J Pattern Recognit Artif Intell 17(6):903 929
10. Kim Y, Nick Street W, Menczer F (2006) Optimal ensemble construction via meta evolutionary ensembles. Expert Syst Appl 30(4):705 714
11. Muthuramalingam A, Himavathi S, Srinivasan E (2007) Neural network implementation using FPGA: Issues and application. Int J Inf Technol 4(2):86 92
12. Dietterich TG (1997) Machine learning research: four current directions. AI Mag 18(4):97 136
13. Optiz DW (1999) Feature selection for ensembles. Proceedings of the 16th International Conference on Artificial Intelligence, pp 379 384
14. Bryll F, Gutierrez Osuna R, Quek F (2003) Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognit 36(6):1291 1302
15. Kolen JF, Pollack JB (1991) Back propagation is sensitive to initial conditions. Proceedings of the 1990 Conference on Advances in Neural Information Processing Systems, pp 860 867
16. Friedman J (1996) Another approach to polychotomous classification. Technical report, Stanford University
17. Hastie T, Tibshirani R (1998) Classification by pairwise coupling. Ann Stat 26(2):451 471
18. Dietterich TG, Bakiri G (1995) Solving multiclass learning problems via error correcting output codes. J Artif Intell Res 2:263 286
19. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques. Morgan Kaufmann, Washington
20. Sesmero MP, Alonso Weber JM, Gutiérrez G, Ledezma A, Sanchis A (2007) Testing feature selection in traffic signs. In Proceedings of the 11th International Conference on Computer Aided Systems, pp 396 398
21. Hall MA (1998) Correlation based feature selection for machine learning. Ph.D diss. Department of Computer Science, Waikato University, Hamilton, NZ
22. Xu L, Yan P, Chang T (1998) Best first strategy for feature selection. 9th Int. Conf. On Pattern Recognition, pp 706 708
23. Russell SJ, Norvig P (2003) Artificial intelligence: a modern approach. Prentice Hall, Englewood
24. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing: explorations in the microstructure of cognition, vol 1. MIT, Cambridge
25. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123 140
26. Quinlan JR (1996) Bagging, boosting, and C4.5. Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, OR
27. Opitz D, Maclin R (1999) Popular ensemble methods: an empirical study. J Artif Intell Res 11:169 198
28. Duin R, Tax D (2000) Experiments with classifier combining rules. Proceedings of the first international workshop on multiple classifier systems. Lect Notes Comput Sci 1857:16 29
29. Tsymbal A, Pechenizkiy M, Cunningham P (2003) Diversity in ensemble feature selection. Technical report TCD CS 2003 44, Computer Science Department Trinity College, Dublin
30. Kuncheva LI, Witaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51(2):181 207
31. Alpaydin E (1999) Combined 5×2cv F test for comparing supervised classification learning algorithms. Neural Comput 11:1885 1892
32. Sesmero MP, Alonso Weber JM, Gutiérrez G, Ledezma A, Sanchis A (2007) Specialized ensemble of classifiers for traffic sign recognition. Comput Ambient Intell Lect Notes Comput Sci 4507:733 740
33. Ship CA, Kuncheva LI (2002) Relationships between combination methods and measures of diversity in combining classifiers. Inf Fusion 3(2):135 148
34. LeCun Y, Jackel LD, Bottou L, Cortes C, Denker JS, Drucker H, Guyon I, Muller UA, Sackinger E, Simard P, Vapnik V (1995) Learning algorithms for classification: a comparison on handwritten digit recognition. In: JH Oh, Kwon C, Cho S (eds) Neural networks: the statistical mechanics perspective. World Scientific, Singapore, pp 261 276
35. Everitt BS (1977) The analysis of contingency tables. Chapman and Hall, London
36. Sharkey AJC, Sharkey NE, Gerecke U, Chandroth GO (2000) The "test and select" approach to ensemble combination. Lect Notes Comput Sci 1857:30 44
37. Roli F, Giacinto G, Vernazza G (2001) Methods for designing multiple classifier systems. Lect Notes Comput Sci 2096:78 87
38. Goebel K, Yan W (2004) Choosing classifiers for decision fusion. Proceedings of the Seventh International Conference on Information Fusion, vol 1, pp 563 568
39. Ledezma A, Aler A, Sanchis A, Borrajo D (2010) GA stacking: evolutionary stacked generalization. Intell Data Anal 14(1):89 119
40. Schapire RE (1990) The strength of weak learnability. Mach Learn 5(2):197 227