Information Fusion 12 (2011) 111–130

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

An empirical study of binary classifier fusion methods for multiclass classification

Nicolás García-Pedrajas *, Domingo Ortiz-Boyer

Department of Computing and Numerical Analysis, University of Córdoba, Spain

ARTICLE INFO

Article history:
Received 6 October 2009
Received in revised form 7 June 2010
Accepted 25 June 2010
Available online 3 July 2010

Keywords:
Multiclass problems
Class binarization
Output coding
Classification

ABSTRACT

One of the most important topics in information fusion is the combination of individual classifiers in multi-classifier systems. We have two different tasks in this area: one is the training and construction of ensembles of classifiers, with each one being able to solve the multiclass problem; the other task is the fusion of binary classifiers, with each one solving a different two-class problem to construct a multiclass classifier. This paper is devoted to the study of several aspects on the fusion process of binary classifiers to obtain a multiclass classifier.

In the general case of a classification problem with more than two classes, we are faced with the issue that many algorithms either work better with two-class problems or are specifically designed for two-class problems. In such cases, a binarization method that maps the multiclass problem into several two-class problems must be used. In this task, information fusion plays a central role because of the combination of the prediction of the different binary classifiers into a multiclass classifier. Several issues regarding the way binary learners are trained and combined are raised by this task. Issues such as individual accuracy, diversity, and independence are common to other information fusion tasks such as the construction of ensembles of classifiers.

This paper presents a study of the different class binarization methods for the various standard multiclass classification problems that have been proposed while addressing aspects not considered in previous works. We are especially concerned with many of the general assumptions in the field that have not been fully assessed by experimentation. We test the different methods in a large set of real-world problems from the UCI Machine Learning Repository, and we use six different base learners.

Our results corroborate some of the previous results present in the literature. Furthermore, we present new results regarding the influence of the base learner on the performance of each method. We also show new results on the behavior of binary testing error and the independence of binary classifiers depending on the coding strategy. Finally, we study the behavior of the methods when the number of classes is high and in the presence of noise.

1. Introduction

A supervised classification problem of K classes and n training observations consists of a set of patterns whose class membership is known. Let $T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ be a set of n training samples where each instance $\mathbf{x}_i$ belongs to a domain X. Each label is an integer from the set $Y = \{1, \ldots, K\}$. A multiclass classifier is a function f: $X \rightarrow Y$ that maps an instance $\mathbf{x}$ into an element of Y.

The task is to find a definition for the unknown function $f(\mathbf{x})$ given the set of training instances. Although many real world problems are multiclass problems, $K > 2$, many of the most popular classifiers work best when facing two class problems, $K = 2$. Indeed many algorithms are specifically designed for binary problems, such as support vector machines (SVMs) [1]. A class binarization is the mapping of a multiclass problem into several two-class problems in a way that allows the derivation of a prediction for the multiclass problem from the predictions of the two-class classifiers. The two-class classifier is usually referred to as the *binary classifier* or *base learner*.

The main aim of this paper is to discover the factors that are more relevant to the construction of a multiclass classifier from the fusion of many binary classifiers. We must bear in mind that the fusion of binary classifiers is different from other similar tasks, such as the fusion of classifiers in an ensemble [2]. As each classifier solves different problems, there are hard constraints [3] in the way the classifiers can be combined. Similarly, classifier construction is constrained by the binary problem, and most of the methods, such as diversity promotion [4] or classifier selection [5], are not applicable.

There are three basic approaches to multiclass problems using base learners that are only able to discriminate between two classes:

* Corresponding author.
E-mail addresses: npedrajas@uco.es (N. García-Pedrajas), dortiz@uco.es (D. Ortiz-Boyer).

1566-2535/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.inffus.2010.06.010

- *One-vs-all.* The one-vs-all method has been proposed by several authors independently [6,7]. This method constructs K binary classifiers. The ith classifier, f_i, is trained using all the patterns of class i as positive instances; the patterns of the other classes are negative instances. An example is classified in the class whose corresponding classifier has the highest output.
- *One-vs-one.* The one-vs-one method proposed in [8] constructs $K(K-1)/2$ classifiers [9]. Classifier ij, named f_{ij}, is trained using all the patterns from class i as positive instances; all of the patterns from class j are negative instances, and the rest are disregarded. There are different methods of combining the obtained classifiers. The most common method is a simple voting scheme. When classifying a new instance, each of the base classifiers cast a vote for one of the two classes used in its training. Other more sophisticated methods have also been proposed [10–12].

  Another approach for the combination of trained classifier is the *decision directed acyclic graph* (DDAG) [10]. This method constructs a rooted binary acyclic graph using the trained classifiers as for the one-vs-one method. The nodes are arranged in a triangle with the root node at the top, two nodes in the second layer, four in the third layer, and so on. To evaluate a DDAG on input pattern $\mathbf{x}$, the binary function is evaluated starting at the root node; the next node visited depends upon the results of this evaluation. The final answer is the class assigned by the leaf node visited at the final step. This method will not be considered in our experiments, as its results are highly similar to the fusion produced by voting classifiers.
- *Error-correcting output codes (ECOC).* Dietterich and Bakiri [13] suggested the use of error-correcting codes for multiclass classification. This method uses a matrix M of $\{-1,1\}$ values of size $K \times F$, where F is the number of binary classifiers. The jth column of the matrix induces a partition of the classes into two *metaclasses*. Instance $\mathbf{x}$ belonging to class i is a positive instance for the jth classifier if and only if $M_{ij} = 1$. If we designate f_j as the sign of the jth classifier, the decision implemented by this method, $f(\mathbf{x})$, using the Hamming distance between each row of the matrix M and the output of the F classifiers is given by

$$f(\mathbf{x}) = \arg\min_{r\in 1,\dots,K} \sum_{i=1}^{F} \left(\frac{1 - \operatorname{sign}(M_{ri} f_i(\mathbf{x}))}{2} \right). \quad (1)$$

There are other approaches that use other distance measures or more sophisticated methods [14]. The usefulness of this approach relies heavily on the independence of the classifiers [15], without which the error-correcting approach would fail. Rifkin and Klautau [16] suggested that if the binary classifiers are fine-tuned more carefully the independence of the classifiers diminishes, as does the efficiency of this approach.

Other fusion methods have been proposed, such as the combination of one-vs-all and one-vs-one [17,12]. However, these are based on the three basic methods. Thus, we will limit our study to the features of these three basic fusion methods.

Allwein et al. [18] proposed a unifying approach for the different methods using a coding matrix with three values, $\{-1,0,1\}$, with 0 meaning *don't care*. For example, in the one-vs-all approach, the coding matrix has K columns, all the diagonal elements are set to 1, and all other elements are set to -1. For one-vs-one, there is a matrix of $K \times \binom{K}{2}$, in which each column corresponds to a pair (c_1, c_2). For this column, the matrix has +1 in row c_1, -1 in row c_2, and zeros in all other rows. In the same work, Allwein et al. introduced *sparse ECOC codes*, which allow 0 values in the coding matrix. In their experiments, each bit of the matrix had an equal probability of having a non-zero value, -1 or +1, and a 0 value. Standard codes were named *dense codes*.

From a practical point of view, the preferred approach is not yet established. Although most previous works [18,19] agree that the one-vs-one and ECOC methods are superior to one-vs-all methods, a recent work [16] has raised serious doubts about the conclusiveness of this assumption. This work [16] claims that if the base learners are properly tuned, most of the reported differences in terms of testing error disappear.

In this work we report a comprehensive set of experiments that aim to evaluate the performance of these fusion methods as accurately as possible. Our conclusions are supported by a large set of problems, including 44 datasets for the one-vs-one and one-vs-all methods and 30 datasets for the ECOC method. There are also six different base learners: neural networks, decision trees, support vector machines with a linear and a Gaussian kernel, logistic regression and naive Bayes classifiers. With this large set of problems we have performed many different experiments to assess the weaknesses and strengths of each method.

The experiments reported in this paper will result in various conclusions. We report experimental evidence that the performance of ECOC codes improves as longer codewords are added, but the improvement is less dramatic for longer codes. In general, no significant differences between random codes and codes designed specifically for their error-correcting capabilities are found. Dense and sparse codes exhibit the same behavior in absence of noise. However when noise is added, the dense codes are slightly more robust than the sparse codes. As a general rule, one-vs-one method is the method most affected by noise, which degrades its performance considerably. Similar results are obtained when powerful learners are used. We will also report a comparison against native multiclass methods, showing that ECOC codes generally perform better than native multiclass methods. The report will demonstrate the usefulness of the fusion of binary classifiers, as they outperform a monolithic classifier even for powerful learners. We will also show that the binary testing error is deteriorated when the codewords are designed by their error-correcting capabilities, a conclusion that has been suggested repeatedly but has never been thoroughly proved. Also, the independence of classifiers is improved by designing the codes for their error-correcting capabilities, especially with sparse codes. This too is a conclusion usually taken for granted but never tested with as many experiments as we present.

It should be note that even when our results corroborate previous results, they are still interesting in that our experimental setup is more robust than those used in previous works. Our two first two main contributions, then, will be the number of datasets used and the number of base learners studied. Additionally, the following new results will be presented in this paper:

- An extensive comparison of the influence of code length on the performance of the error-correcting methods. This comparison shows that the testing error decreases with longer codes, but the reduction is less significant from 100 bits onwards.
- The same comparison has shown that codes designed by their error correcting capabilities are better than random codes only for small lengths and certain base learners. No previous study has undergone this comparison using the same amount of base learners and datasets.
- Sparse codes are able to improve their performance with codeword length, with a overall better behavior than dense codes for long codewords.
- One-vs-all performs worse than the other methods with the exception of using a support vector machine with a Gaussian kernel. This result explains previous conflicting experiments in the literature.

- For powerful learners, ECOC codes are the best choice. However, for simpler learners, one-vs-one is the best algorithm.
- The binary testing errors in the ECOC codes are worse when the codes are designed for their error-correcting capabilities. However, the binary classifiers are also more independent.
- The relative performance of the methods does not depend on the number of classes in the datasets.
- Noise affects one-vs-one and one-vs-all more than error correcting codes.

Furthermore, an extensive parameter setting process has been performed for all learners, as the impact of parameters in comparing different methods has been shown to be decisive [16].

This paper is organized as follows: Section 2 discusses related works; Section 3 shows the experiment design; Section 4 explains the experimental setup; Section 5 shows the results of the experiments; and Section 6 states the conclusions of our work.

2. Related work

Various researches have made comparative studies of the performance of different multiclass output coding schemes. Nevertheless, most of these studies are not conclusive and some of the claims are not justified enough by the experiments as studied by Rifkin and Klautau. In [16] they summarized results of previous works comparing different coding methods and found that in many of them the reported differences were either too small to be significant or due to bad parameter setting. They refuted the conclusions of many previous works with a careful experiment design.

Allwein et al. [18] presented a theoretical study of output coding where a comparison among one-vs-all, one-vs-one and different ECOC codes was carried out. The main conclusion was that one-vs-all method was worse than the rest. Nevertheless, many of the error results for one-vs-all can be greatly improved just by choosing a better parameter set [16], so their conclusions must be taken with caution.

Although the original insight of Dietterich and Bakiri was that the error-correcting properties of the codes should be able to correct mistakes and improve classification, in practical applications the correlation among the base learners may render the large distances between vectors useless [20]. Most previous papers agreed that for ECOC codes, obtaining binary classifiers that are uncorrelated [21] is a key factor. Kong and Dietterich [21] showed that ECOC method was able to improve bias and variance of the error, and suggested that bias reduction was achieved making use of the non-local properties of the learning algorithm. Thus, they hypothesized that local learning algorithms, such as k-NN and RBF networks, will not be able to benefit from this method.

Hsu and Lin [22] performed a comparison of one-vs-one, one-vs-all, DDAG, and two multiclass native methods[1] for SVM. Although they concluded that one-vs-all is inferior to one-vs-one, the experiments are not so conclusive as only ten datasets are considered, and only two of them have more than seven classes. Furthermore, for seven problems, all the methods resulted in an error below 5%, leaving little room for significant improvement.

In a set of experiments limited to a reduced set of real-world problems, Windeatt and Ghaderi [14] found that equidistant codes achieved a better performance than random codes, but the differences in performance were reduced as the length of the codes increased, and the authors' conclusion was that more work was needed to test this hypothesis.

[1] We will use the term "native multiclass method" to refer to a multiclass method that is inherent to a certain classifier and does not use any of the coding methods studied in this paper.

More recently, Rifkin and Klautau [16] showed that one-vs-all strategy was able to perform, at least, as well as the other methods using a SVM with a Gaussian kernel as base learner and tuning the relevant parameters of the learning algorithm, C and γ, appropriately. Similar behavior of one-vs-all and SVMs was reported in Passerini et al. [23]. Rennie and Rifkin [24] argued that the performance of output coding methods largely depends on binary classifier performance. They supported this idea with results from multiclass text classification.

Masulli and Valentini [25] made a study of several aspects of error-correcting codes. They found that decoding schemes based on L^1 and L^2 norms were able to outperform Hamming decoding. A further study was made by the same authors [26], taking into account relationships between the error recovering power, the accuracy of the base learners, and the dependence among codewords bits. This paper concluded that the relationship between these features very likely depends on the specific dataset, as no general trends could be found.

Many researchers [13,27] agree that random codes are a good alternative for error-correcting codes, and that more sophisticated methods might have only marginal effects on testing error. Moreover, as the length of the codewords increases, the overall performance on a set of problems of different coding and decoding methods tends to be very similar. Furthermore, in [28] it is claimed that a long enough random code would not be outperformed by a predefined code.

Recent works have shown that ECOC methods can improve their performance if code design is guided by the training data. Pujol et al. [29] developed Discriminant ECOC, where the coding matrices are based on an hierarchical partition of the class space which is performed by means of a maximization of a discriminative criterion. The same authors also proposed a constructive approach for obtaining the coding matrix called ECOC-ONE [30]. Escalera et al. [31] constructed the coding matrix focusing on the binary problems that best fitted the decision boundaries of the data. Melvin et al. [32] improved one-vs-all classification by means of a perceptron used to learn a weighting of the binary classifiers.

3. Designing the comparison

Our work aims to study as many different aspects of the fusion of binary classifiers in output coding as is relevant, from the point of view of the performance of the classifier. This is not an easy task, as previous works have shown that there are many issues to be taken into account, and the relationships among them are highly complex. Several decisions must be made to make a useful comparison among the algorithms.

The first decision is which base learner to use. It is an established fact [16] that if weak learners are used, the errors of those classifiers tend to be more independent. Consequently, certain types of coding methods, such as one-vs-one and ECOC, may perform better, as the fusion of these classifiers benefits more from the independence of the errors. On the other hand, more powerful classifiers will have less independent errors. Consequently, the methods that do not rely on the independence of the binary classifier, such as one-vs-all, may benefit.

Thus, the selection of the binary classifier to be used has a major impact on the experimental results [26]. Taking this fact into account, we have chosen six base learners. The idea behind using six base learners instead of just one is to study whether the type of base learner is relevant to the comparison of the output coding methods. As each binary classifier has its own learning bias, the fusion of different kinds of classifiers may give raise to different issues. Thus, by using different binary classifiers we try to account for the different biases and their effects on the combination process. The chosen methods are a neural network trained with a

back-propagation algorithm, a C4.5 decision tree, logistic regression, a naive Bayes classifier (NBC) and support vector machines (SVMs) using linear and Gaussian kernels. The SVMs with these two kernels are considered separate classifiers as their behaviors are quite different depending on the kernel. These are several of the most widely used methods for classification in real-world problems.

We must also decide upon the decoding scheme for the ECOC codes. Choosing the best decoding scheme is an open question as there is no general agreement on which decoding method is the best, if any. The most commonly used methods are the Hamming and L^1 distances. However, in a comparison of decoding strategies, Windeatt and Ghaderi [14] did not find significant differences among these strategies for both artificial and natural dataset. Therefore, we have chosen the L^1 norm as the decoding strategy. For a coding matrix M of $\{-1,1\}$ values of size $K \times F$ where F is the number of binary classifiers, f_j designates the output of the jth classifier, the decision implemented using L^1 norm assigns an instance $\mathbf{x}$ to the class y given by:

$$y = \arg\min_{r\in 1,\dots,K} L^1(\mathbf{f}(\mathbf{x}) - M_r) = \arg\min_{r\in 1,\dots,K} \sum_{i=1}^{F} |f_i(\mathbf{x}) - M_{r,i}|. \quad (2)$$

4. Experimental setup

For the comparison of the different models we selected 44 datasets from the UCI Machine Learning Repository. A summary of these datasets is shown in Table 1. The datasets were selected considering problems of at least six classes for the ECOC codes, and of at least three classes for the other methods. For all methods, ties were broken by assigning the query instance to the most frequent class; if the tied classes had equal frequency, we assign the class randomly among them.

The experiments were conducted following the 5×2 cross-validation setup [33]. We performed five replications of a twofold cross-validation. In each replication the available data were partitioned into two random, equal-sized sets. Each learning algorithm was trained on one set at a time and then tested on the other set.

We used the Wilcoxon test for comparing pairs of algorithms. The formulation of the test [34] is as follows: let d_i be the difference between the error values of the methods on the ith dataset. These differences are ranked according to their absolute values; in case of a tie an average rank is assigned. Let R^+ be the sum of the ranks for the datasets on which the second algorithm has outperformed the first, and let R^- be the sum of ranks where the first algorithm has outperformed the second. The ranks of $d_i = 0$ are split evenly among the sums:

$$R^+ = \sum_{d_i>0} \mathrm{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \mathrm{rank}(d_i), \quad (3)$$

and,

$$R^- = \sum_{d_i<0} \mathrm{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \mathrm{rank}(d_i). \quad (4)$$

Let T be the smaller of the two sums and N the number of datasets. For a small N, there are tables with the exact critical values for T. For a larger N, the statistic

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \quad (5)$$

is distributed approximately according to $N(0,1)$. The Wilcoxon test was chosen because it assumes limited commensurability. This test is also safer than parametric tests, because it does not assume normal distributions or a homogeneity of variance. Furthermore, empirical results [35] show that it is stronger than other tests. The p-value of the test is shown in all the tables. When the p-value is below 0.05, we can conclude at a confidence level of 95% that there are significant differences between the two compared methods.

However, when groups of algorithms are considered, it is advisable to perform a multiple comparison test first. Therefore, when comparing groups of algorithms, we applied an Iman–Davenport test [35]. If significant differences were found, we then used the Wilcoxon test. The Iman–Davenport test is applied to the subset of problems for which all the tested algorithms can be applied. The Iman–Davenport test is based on the χ_F^2 Friedman test [36], which compares the average ranks of k algorithms. Let r_i^j be the rank of the jth algorithm on the ith dataset where in the case of ties average ranks are assigned, and let $R_j = \frac{1}{N}\sum_i r_i^j$ be the average rank for N datasets. Under the null hypothesis, all algorithms are equivalent, the statistic:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \quad (6)$$

is distributed following a χ_F^2 with $k-1$ degrees of freedom for k and N that are sufficiently large. In general $N > 10$ and $k > 5$ is enough. Iman and Davenport [37] found that this statistic is too conservative, and they developed a better one:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \quad (7)$$

which is distributed following an F distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. We show in the tables the p-value of the test and use a confidence value of 95%.

Rifkin and Klautau [16] showed the impact of the parameter setting on the performance of algorithms. They reported large differences for the same algorithm depending on the parameter setting, using a SVM as base learner. Thus, we performed a preliminary set of experiments to test the sensitivity of the base learners to the parameter settings. We tested the influence of the adjustable parameters of the C4.5 algorithm, and we found no significant differences. Using the source code provided by Quinlan,[2] we had three parameters: the decision of whether to use pruned or unpruned trees; the number of trials to construct the trees; and whether to soften the thresholds. We used pruned trees, one trial and the softening of thresholds.

For the remaining base learners, the experiments showed from moderate to marked differences depending on the parameters. Thus, for these base learners, we performed a fivefold cross-validation of the parameters. Each time a base learner was trained, we got five random folds of the training data and we tested a fixed set of parameters.

For neural networks, we chose a multi-layer perceptron with 3, 10 and 25 hidden nodes; 5000 and 15,000 epochs of eight randomly selected instances; a learning rate, η, of 0.15 and 0.5; and a momentum term, α, of 0.0 and 0.001. We then tested the 24 different combinations of parameters. For a SVM with a linear kernel we tried $C \in \{0.1, 1, 10\}$, and for a Gaussian kernel we tried $C \in \{0.1, 1, 10\}$ and $\gamma \in \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, thereby testing all 18 combinations.

The logistic regression had two relevant parameters to be set: the learning rate, η, and the regularization parameter, λ. We tried $\eta \in \{0.001, 0.01, 0.5\}$ and $\lambda \in \{0.0, 0.00001, 0.0001, 0.001, 0.01\}$, testing all 15 combinations. For NBC, as we were dealing with continuous inputs in most cases, the probabilities had to be calculated for the intervals of the input values. The number of intervals was ad-

[2] Available at http://www.rulequest.com/Personal/c4.5r8.tar.gz.

Table 1
Summary of data sets. The features of each data set can be C (continuous), B (binary) or N (nominal). The number of nontrivial dichotomies available is shown for each type of output coding method: Dense ECOC, sparse ECOC, and one-vs-one.

Data set	Cases	Classes	Features			Inputs	Binary classifiers		
			C	B	N		Dense ECOC	Sparse ECOC	one-vs-one
Abalone	4177	29	7	–	1	10	2.68e+8	3.43e+13	406
Anneal	898	5	6	14	18	59	15	90	10
Arrhythmia	452	13	279	–	–	279	4095	7.88e+5	78
Audiology	226	24	–	61	8	93	8.38e+6	1.41e+11	276
Autos	205	6	15	4	6	72	31	301	15
Balance	625	3	4	–	–	4	3	6	3
Car	1728	4	–	–	6	16	7	25	6
Dermatol.	366	6	1	1	32	34	31	301	15
Ecoli	336	8	7	–	–	7	127	3025	28
Gene	3175	3	–	–	60	120	3	6	3
Glass	214	6	9	–	–	9	31	301	15
Horse	364	3	7	2	13	58	3	6	3
Hypo	3772	4	7	20	2	29	7	25	6
Iris	150	3	4	–	–	4	3	6	3
Isolet	7797	26	34	–		617	3.35+7	1.27e+12	325
Krkopt	28,056	18	6	–	–	6	1.31e+5	1.93e+8	153
Led24	200	10	–	24	–	24	511	28,501	45
Letter	20,000	26	16	–	–	16	3.35e+7	1.27e+12	325
Lrs	531	10	101	–	–	101	511	28,501	45
Lymph	148	4	3	9	6	38	7	25	6
Mfeat-fac	2000	10	216	–	–	216	511	28,501	45
Mfeat-fou	2000	10	76	–	-	76	511	28,501	45
Mfeat-kar	2000	10	64	–	–	64	511	28,501	45
Mfeat-mor	2000	10	6	–	–	6	511	28,501	45
Mfeat-pix	2000	10	240	–	–	240	511	28,501	45
Mfeat-zer	2000	10	47	–	–	47	511	28,501	45
New-thyroid	215	3	5	–	–	5	3	6	3
Nursery	12,960	5	–	1	7	23	15	90	10
Optdigits	5620	10	64	-	–	64	511	28,501	45
Page-blocks	5473	5	10	–	–	10	15	90	10
Pendigits	10,992	10	16	–	–	16	511	28,501	45
Primary-t.	339	22	–	14	3	23	2.09e+6	1.56e+10	231
Satimage	6435	6	36	–	–	36	31	301	15
Segment	2310	7	19	–	–	19	63	966	21
Shuttle	58,000	7	9	–		9	63	966	21
Soybean	683	19	-	16	19	82	2.62e+5	5.80e+8	171
Texture	5500	11	40	–	–	40	1023	86,526	55
Vehicle	846	4	18	–	–	18	7	25	6
Vowel	990	11	10	–	-	10	1023	86,526	55
Waveform	5000	3	40	–	–	40	3	6	3
Wine	178	3	13	–	–	13	3	6	3
Yeast	1484	10	8	–	–	8	511	28,501	45
Zip	9298	10	256	–	–	256	511	28,501	45
Zoo	101	7	1	15	–	16	63	966	21

justed using the same cross-validation method described above, and values of $n = \{5, 10, 25, 50, 100\}$ were used. Together with the number of intervals we also tested four methods for obtaining the interval bounds [38]: equal width discretization (EWD); equal frequency discretization (EFD); proportional k-interval discretization (PKID); and the Fayyad and Irani method [39]. These methods gave us a total of 16 combinations of parameters.

The source code used for all methods is licensed under the GNU General Public License and is freely available upon request from the authors. The SVM was implemented using `libsvm` [40] library.

We did not considered the execution time in the comparison of the algorithms. Although important, a comparison of speed not only measures the speed of an algorithm, but also the efficiency of a specific implementation, making the results difficult to extrapolate if better implementations can be achieved. Fürkranz [19] showed theoretically that one-vs-one classifiers are less complex than one-vs-all classifiers. Nevertheless, specific implementations can be made that overcome these different levels of complexity. For instance, when using a SVM as a base classifier, the complexity of one-vs-all can be reduced by reusing the support vectors [16].

5. Experimental results

In this section we present the results of the experiments performed following the experimental setup above. We have devoted each section to a specific study. Specific conclusions are summarized at the end of each section. After each experimental conclusion, we include a short summary of previous results related to that conclusion. In each summary, only works that have included a sufficient number of datasets are considered. Detailed results for each experiment are available upon request from the authors.

5.1. Error-correcting output codes

In the first stage, we studied different aspects that affect the performance of ECOC methods. This first set of experiments had two objectives: (i) to study the effect of the number of bits of the codewords on the testing error of the classifier; and (ii) to study the effect of the error-correcting capabilities of the codes on the testing error. This work was done for dense and sparse codes.

To study the effect of the number of bits, we trained the base learners with codes of 30, 50, 100, and 200 bits. The coding

matrices were obtained randomly with an approximately equal split between +1's and −1's. The random codes were selected without allowing repeated columns or repeated rows.

Many researchers [13,27] agree that random codes are a good alternative for error-correcting codes, and that more sophisticated methods may have only marginal effects on testing error. Moreover, as the length of the codewords increases, the overall performance on a set of problems of different coding and decoding methods tends to be quite similar. For this reason, it is of interest to study the comparison of random codes and codes designed for their error correcting capabilities. Different methods have been proposed to obtain ECOC codes, such as randomized hill climbing [13], BCH codes [41], simulated annealing and evolutionary computation [42,43]. We tested these methods and found that the best results, in terms of column and row separation, were obtained using evolutionary computation. Moreover, evolutionary computation [44] methods are able to jointly optimize row and column separation.

For the evolution of the population, we used the CHC algorithm [45]. This algorithm optimizes row and column separation. We will refer to these codes as CHC codes throughout the paper for the sake of brevity. The evolution used a population of 100 individuals that was evolved for 10,000 generations, with a mutation rate of 10% and a bit mutation rate of 10%.

In all of the comparison tables we show the win/draw/loss record (s) of the algorithm in the column against the algorithm in the row. We also show the p-value of the Wilcoxon test (p_w). The win/draw/loss record reports how many times the algorithm in the column was better than the algorithm in the row (win), was equal (draw), or was worse (loss). For every pair of algorithms, the comparison was made with the results of the problems to which both methods could be applied. As an additional test, we show the p-value of a sign test on the win/loss record (p_s).

5.1.1. Comparison of codewords length and error-correcting capabilities

For problems with seven classes, we tested random and CHC codes up to 50 bits. For problems with eight classes we tested both codes up to 100 bits. For problems with more than eight classes all codes (30, 50, 100 and 200 bits) were used. Tables 2 and 3 show the comparison of the different codes. Each table shows the results of the Iman–Davenport test over all the methods.

For a neural net as a base learner we can see that for both random and CHC codes, the generalization ability of the classifier improves as more bits are added. The differences are significant for the comparisons between all of the different lengths (p-values of the Wilcoxon test below 0.05 for all lengths, as shown in Table 2). The results of the comparison between random and CHC codes confirm previous works that state that random codes are no worse than codes designed for their error-correcting capabilities. CHC codes are better than random codes only for 30 bits, with a p-value of 0.0407. All of the remaining comparisons show p-values above 0.05, 0.2087 for 50 bits; 0.1283 for 100 bits; and 0.5373 for 200 bits. This lack of significant differences between random and CHC codes is especially noticeable if we take into account the fact that we obtained the random codes using the simplest approach. The only imposed constraint was avoiding repeated rows and columns and trivial dichotomies.

Using C4.5 as base learner obtained nearly identical results, but the CHC codes were not able to beat random codes for any length. All the p-values were above the critical level: 0.1779 for 30 bits; 0.9393 for 50 bits; 0.9032 for 100 bits; and 0.1677 for 200 bits. Again, the performance of the codes was better as the length of the code increased. As stated, the comparison for the SVM as a base learner was made separately for the linear and Gaussian kernels. For a linear kernel we found significant differences between random and CHC codes for small lengths of 30 and 50 bits, with p-values of 0.0460 and 0.0153, respectively; however, these differences disappeared with longer codes. The 200-bit CHC codes were worse than random ones, with a p-value of 0.0262. This trend of worse performance of CHC codes for longer lengths was found in all of the less powerful base learners (SVM with a linear kernel, NBC and the logistic regression). In Section 5.3 we study the binary testing error of the learners. This section shows that the binary errors for CHC codes are worse than for random codes. This fact may explain why the less powerful learners are affected by those with more difficult dichotomies, which may be why long random codes are better than CHC codes of the same length.

For a Gaussian kernel, we obtained a better performance with the longer codes, and the differences between the random and CHC codes were not significant for any length with p-values as follows: 0.4165 for 30 bits; 0.4461 for 50 bits; 0.0830 for 100 bits; and 0.4455 for 200 bits. When using logistic regression, we found the same behavior as for a SVM with a linear kernel. However, a different behavior was found for the comparison of random and CHC codes. In this case, the random codes were no worse than the CHC codes even for small lengths, with the exception of 50-bit codes, which had a p-value of 0.0155. For 200-bit codes, the random codes were significantly better than the CHC codes, as explained in the previous paragraph. Finally, using NBC as base learner, we observed the same pattern of behavior. The CHC codes were better for small codes, but worse for longer codes, though not significantly.

As a general conclusion, the overall behavior of the algorithm does not differ much depending on the base learner. Similar conclusions can be extracted from the six comparison tables. For random codes we see an increasing performance as the length of the code increases. The increase is less dramatic as the code becomes longer. For CHC codes, the situation is similar. However, there is a difference regarding CHC codes, as the weaker learners are able to take better advantage of error-correcting capabilities than the stronger learners. Furthermore, the behavior within each of the two groups—i.e., the powerful base learners (neural networks, decision trees and SVMs with a Gaussian kernel) and the less powerful learners (SVMs with a linear kernel, logistic regression and NBC)—is even more homogeneous.

Experimental conclusion 1. Codeword length. The testing error decreases with longer codes. The improvement is less dramatic as the length of the code is greater, but there are still significant differences between the 100- and 200-bit lengths.
Previous results. Windeatt and Ghaderi [14] performed a comparison of codes of different lengths but with a longest code of 31 bits. Only five real-world datasets where used with a maximum of seven classes.
Experimental conclusion 2. Codeword design. Codes designed by their error-correcting capabilities are better than random codes only for small lengths and certain base learners.
Previous results. Windeatt and Ghaderi [14] performed a comparison of equidistant and random codes for five small real-world problems. However, no experiments were performed that took into account column separation.
In the previous experiments we have shown that the error-correcting capabilities of the codes are unable to improve the performance of the codes as a whole. This result prompts a new experiment. In the previous experiment, we designed the ECOC codes using row and column separation; in the new experiment, we consider codes designed using distinct column and row separation to study which one is more relevant. Table 4 shows a comparison using codes of 100 bits. The results show a clear trend toward row separation. It appears that the codes designed using only row separation performed almost as well

Table 2
Comparison of results for ECOC in terms of testing error using neural networks, decision trees and a SVM with a linear kernel.

		30 bits		50 bits		100 bits		200 bits	
		Random	CHC	Random	CHC	Random	CHC	Random	CHC
Neural networks									
Iman–Davenport test p-value: 0.0000									
Random	s		21/0/9	25/0/1		22/0/1		22/0/0	
30	p_w		0.0407	0.0000		0.0000		0.0000	
CHC	s				24/0/2		23/0/0		22/0/0
30	p_w				0.0000		0.0000		0.0000
Random	s				16/0/10	22/0/1		22/0/0	
50	p_w				0.2087	0.0000		0.0000	
CHC	s						22/0/1		22/0/0
50	p_w						0.0000		0.0000
Random	s						14/0/9	20/0/2	
100	p_w						0.1283	0.0003	
CHC	s								18/0/4
100	p_w								0.0058
Random	s								9/1/12
200	p_w								0.5373
C4.5									
Iman–Davenport test p-value: 0.0000									
Random	s		18/0/12	24/0/2		23/0/0		22/0/0	
30	p_w		0.1779	0.0000		0.0000		0.0000	
CHC	s				26/0/0		23/0/0		21/0/1
30	p_w				0.0000		0.0000		0.0000
Random	s				12/1/13	23/0/0		22/0/0	
50	p_w				0.9393	0.0000		0.0000	
CHC	s						22/0/1		21/0/1
50	p_w						0.0002		0.0000
Random	s						11/1/11	19/0/3	
100	p_w						0.9032	0.0006	
CHC	s								17/0/5
100	p_w								0.0081
Random	s								7/0/15
200	p_w								0.1677
SVM linear kernel									
Iman–Davenport test p-value: 0.0000									
Random	s		21/0/9	25/0/1		23/0/0		22/0/0	
30	p_w		0.0460	0.0000		0.0000		0.0000	
CHC	s				25/0/1		23/0/0		22/0/0
30	p_w				0.0000		0.0000		0.0000
Random	s				20/0/6	22/0/1		22/0/0	
50	p_w				0.0153	0.0000		0.0000	
CHC	s						20/0/3		21/0/1
50	p_w						0.0026		0.0000
Random	s						10/1/12	21/0/1	
100	p_w						0.3860	0.0001	
CHC	s								17/1/4
100	p_w								0.0039
Random	s								4/0/18
200	p_w								0.0262

as the codes designed using column and row separation. However, if we only consider column separation, the results are significantly worse.

Experimental conclusion 3. Column and row separation. Codes designed using only row separation perform almost as well as codes designed using column and row separation. On the other hand, codes designed using only column separation are worse.

Previous results. No previous work have compared codes designed separately for row and column distance.

5.1.2. Sparse codes

For sparse codes we had an equal probability of having a 0 in the coding matrix as having a $-1/+1$, which are the values used by Allwein et al. [18]. We performed the set of experiments for sparse codes with the same setup as the dense codes. We then tested random and CHC codes with 30, 50, 100, and 200 bits. For sparse codes the number of nontrivial available dichotomies for problems with six classes is 301 (see Table 1). For this type of codes, then, we performed all of the experiments for all datasets with six classes or more. Tables 5 and 6 show the comparisons.

The behavior of the sparse codes differed in some ways from the behavior of the dense codes. For sparse codes, significant differences were found between random and CHC codes for more lengths. As will be shown in Section 5.4, the CHC's sparse codes were more successful than the dense codes at improving independence among the binary learners. This improvement may be partly responsible for the better behavior of the CHC codes. Regarding the length, we found the same pattern, that is, improved behavior with longer codes. For neural networks, the CHC codes were better than the random codes for 30 bits, with a p-value of 0.0041, and 50 bits, with a p-value of 0.0111. For C4.5, the CHC codes were able to beat the random codes for 30, 50 and 100 bits at a confidence level of 95%. For a SVM with a Gaussian kernel, the CHC codes were significantly better than random ones for all lengths. For the less powerful learners, CHC codes were better than random ones only for 30 bits, with p-values of 0.0687 for a SVM with a linear kernel, 0.0193

Table 3
Comparison of results for ECOC method in terms of testing error using a SVM with a Gaussian kernel, logistic regression and a naive Bayes classifier.

		30 bits		50 bits		100 bits		200 bits	
		Random	CHC	Random	CHC	Random	CHC	Random	CHC
SVM Gaussian kernel									
Iman–Davenport test p-value: 0.0000									
Random	s		20/0/10	22/0/4		21/0/2		20/0/2	
30	p_w		0.4165	0.0003		0.0001		0.0001	
CHC	s				24/1/1		22/0/1		22/0/0
30	p_w				0.0001		0.0000		0.0000
Random	s				14/2/10	19/1/3		21/0/1	
50	p_w				0.4461	0.0007		0.0003	
CHC	s						19/0/4		19/1/2
50	p_w						0.0042		0.0004
Random	s						14/0/9	20/0/2	
100	p_w						0.0830	0.0001	
CHC	s								16/0/6
100	p_w								0.0194
Random	s								11/0/11
200	p_w								0.4455
Logistic regression									
Iman–Davenport test p-value: 0.0000									
Random	s		18/0/12	23/0/3		23/0/0		22/0/0	
30	p_w		0.5038	0.0001		0.0000		0.0000	
CHC	s				26/0/0		23/0/0		22/0/0
30	p_w				0.0000		0.0000		0.0000
Random	s				17/0/9	23/0/0		22/0/0	
50	p_w				0.0155	0.0000		0.0000	
CHC	s						19/0/4		20/1/1
50	p_w						0.0005		0.0001
Random	s						12/0/11	20/1/1	
100	p_w						0.8314	0.0001	
CHC	s								18/0/4
100	p_w								0.0014
Random	s								4/1/17
200	p_w								0.0194
Naive Bayes classifier									
Iman–Davenport test p-value: 0.0000									
Random	s		20/0/10	24/0/2		23/0/0		21/0/1	
30	p_w		0.0545	0.0000		0.0000		0.0000	
CHC	s				22/0/4		22/0/1		21/0/1
30	p_w				0.0001		0.0000		0.0000
Random	s				19/0/7	21/0/2		21/0/1	
50	p_w				0.0694	00.0000		0.0001	
CHC	s						23/0/0		22/0/0
50	p_w						0.0000		0.0000
Random	s						13/0/10	19/0/3	
100	p_w						0.6482	0.0003	
CHC	s								19/0/3
100	p_w								0.0004
Random	s								8/0/14
200	p_w								0.2645

for logistic regression and 0.0001 for NBC. As was the case for dense codes, there was a homogeneous behavior if we considered the classifiers in two separate groups, that is, powerful and less powerful.

Finally, it is interesting to note that our results show that the lengths suggested in [18], $\lceil 10\log_2(k)\rceil$ bits for dense codes and $\lceil 15\log_2(k)\rceil$ bits for sparse codes, are too short, as longer codes are able to significantly improve the results.

Experimental conclusion 4. Sparse ECOC codes. In terms of codeword length and differences between random codes and codes designed by their error-correcting capabilities sparse codes show a similar behavior as dense codes, with the difference of a better general performance of CHC codes.
Previous results. Allwein et al. [18] studied sparse codes but with a maximum length of 71 bits. No comparison between random codes and codes designed for their error-correcting capabilities were performed.

5.1.3. Dense-vs-sparse codes

As an additional test, we compared dense and sparse codes for the codeword lengths and base learners used. The comparison is shown in Table 7. The comparison between the two types of coding matrices did not follow a clear pattern. As a general rule, we can say that as the length of the code increases so does the relative performance of the sparse codes. For a SVM with a linear kernel and logistic regression the sparse codes were better than dense codes for all of the lengths. On the other hand, for a SVM with a Gaussian kernel dense codes outperformed sparse codes. When using a neural network as a base learner, the relative performance depended on the length of the codes. For small lengths the dense codes were better and for longer codes the sparse ones were better. For C4.5 there was a similar behavior. The performance of dense codes deteriorated for longer codes, but sparse codes were unable to equal the performance of dense codes even for 200-bit codes. Using NBC, again we found the same pattern, but there were no

Table 4
Comparison of results for ECOC 100-bit codes using column and row separation in their design separately.

		CHC Rows & columns	CHC Columns	CHC Rows
Neural networks				
Random	s	14/0/9	5/1/17	11/0/12
100	p_w	0.1283	0.0039	0.6265
C4.5				
Random	s	11/1/11	7/0/16	12/0/11
100	p_w	0.9032	0.0208	0.9879
SVM linear kernel				
Random	s	10/1/12	7/1/15	8/1/14
100	p_w	0.3860	0.2360	0.7089
SVM Gaussian kernel				
Random	s	14/0/9	11/1/11	10/1/12
100	p_w	0.0830	0.7210	0.9612
Losgistic regression				
Random	s	12/0/11	8/0/15	8/2/13
100	p_w	0.8314	0.0447	0.6373
Naive Bayes classifier				
Random	s	13/0/10	7/0/16	12/0/11
100	p_w	0.6482	0.0177	0.4645

significant differences between both types of codes for lengths below 50 bits.

The performance of sparse codes with respect to dense codes followed an interesting pattern. More powerful learners showed a better performance for dense codes, and less powerful learners were better for sparse codes. This behavior suggests that as dense codes use all the instances to train the classifier, the induced dichotomies are more difficult to learn, thus more powerful learners are needed to obtain successful results. On the other hand, sparse codes induce simpler dichotomies, as only some of the classes are considered; these simpler dichotomies are successfully dealt with by simpler learners. As will be seen in Section 5.2, this pattern of behavior will also be found in the one-vs-one method.

Experimental conclusion 5. Comparison between dense and sparse codes. Sparse codes improve their relative behavior with respect to dense codes as codeword length increases. The relative performance of dense and sparse codes depends on the base learner used.
Previous results Allwein et al. [18] compared dense and sparse codes but with a maximum length of 47 and 71 bits, respectively.

It may be argued that using the same probability of having a 0 and either a -1 or a +1 may have a bad influence on the performance of sparse coding, as the columns will have too many 0s. To test the influence of theses probabilities we carried out a new set of experiments using codewords of 100 bits. In these experiments, all the three values had the same probability: $P(0) = P(-1) = P(+1) = 1/3$. The comparison with the results of the previous experiments is shown in Table 8. The comparison shows an interesting regular pattern. In general, there were no significant differences between using the two different sets of probabilities, especially for the CHC codes. However, we found differences in some cases. A closer look shows that for base learners where one-vs-one performs well, reducing $P(0)$ worsened the results, which means that more difficult problems are not correctly dealt with by the base learner. Conversely, for the base learners that were not performing well with one-vs-one, the denser codes performed better. The reason may be that columns with less 0's represent more complex problems. Thus, powerful learners can cope with these more complex dichotomies but weaker learners cannot.

5.1.4. Comparison of base learners

As we used six different base learners it is also interesting to test the possible differences in performance among them. We compared their performance for random codewords of 100 bits, which we consider a good compromise between computational cost and performance. The comparison is shown in Table 9 for random dense and sparse codes. The comparison using CHC codes is similar.

The results show that the performance of the three most powerful base learners (neural networks, C4.5 and SVM with Gaussian kernel) was similar, and all of them beat the results obtained using the other three methods. Both the win/draw/loss record and the Wilcoxon test show no significant differences between these three classifiers, as all the p-values are above the critical level of 0.05. This is an expected result if we take into account the *no free lunch* theorem [46]. However, for individual problems there can be significant differences depending on the base learners. Of the three weaker algorithms, NBC had the worst performance, and logistic regression was better than NBC but worse than the SVM with a linear kernel.

The results show that although simple classifiers may benefit from the combination of many classifiers, more powerful learners achieve a better overall accuracy. Therefore, when dealing with real-world problems, powerful base learners are more recommendable than simple ones. However, we must take cautiously any comparison among the different base learners as there are many different algorithms and implementations of most of them, and certain implementations may have a large impact on the performance of a classifier.

Experimental conclusion 6. Differences depending on the base learner. The overall results show a better performance with neural networks, C4.5 and a SVM with a Gaussian kernel. The individual results show large differences.
Previous results. Masulli and Valentini [26] compared different classifiers, but all of them were based on neural networks. No previous works have made comparisons of six different classifiers.

5.2. One-vs-one and one-vs-all methods

The second stage of our work compared the one-vs-one and one-vs-all approaches. For these two methods we used all of the 44 problems described in Table 1. The comparison of the two methods is shown in Table 10 and illustrated in Fig. 1. The figure represents for each point the testing error of the one-vs-one method in the x-axis and the one-vs-all method in the y-axis. Points below the diagonal line show a better performance of one-vs-all, and points above the diagonal line show a better performance of one-vs-one. For a comparison of these two methods with ECOC codes, we considered random dense codes of 100 bits as representative of ECOC codes. For problems with less than 8 classes, for which 100 bits dense codes cannot be obtained, we used the longest available dense code.

The results shows a somewhat clearer picture. One-vs-one was better than one-vs-all using a neural network, C4.5, a SVM with a linear kernel, logistic regression and naive Bayes, with p-values of the Wilcoxon tests of 0.0000 for neural networks, C4.5 and SVM with a linear kernel and of 0.0486 for NBC. On the other hand, one-vs-one was significantly worse than one-vs-all for a SVM using a Gaussian kernel with a p-value of 0.0123.[3] Thus, all these differences were significant at a 95% confidence level. The comparison

[3] This result may explain the results of Rifkin and Klautau [16]. They found that one-vs-all was not worse than one-vs-one, but they used a SVM with a Gaussian kernel as the only base learner.

Table 5
Comparison of results for ECOC sparse codes in terms of testing errors using neural networks, decision trees and a SVM with a linear kernel.

		30 bits		50 bits		100 bits		200 bits	
		Random	CHC	Random	CHC	Random	CHC	Random	CHC
Neural networks									
Iman–Davenport test p-value: 0.0000									
Random	s		25/0/5	29/0/1		29/0/1		30/0/0	
30	p_w		0.0041	0.0000		0.0000		0.0000	
CHC	s				28/0/2		29/1/0		29/1/0
30	p_w				0.0000		0.0000		0.0000
Random	s				20/0/10	28/0/2		28/1/1	
50	p_w				0.0111	0.0000		0.0000	
CHC	s						27/0/3		29/1/0
50	p_w						0.0000		0.0000
Random	s						16/0/14	24/0/6	
100	p_w						0.4779	0.0001	
CHC	s								25/0/5
100	p_w								0.0001
Random	s								12/2/16
200	p_w								0.7577
C4.5									
Iman–Davenport test p-value: 0.0000									
Random	s		25/0/5	29/0/1		29/0/1		30/0/0	
30	p_w		0.0002	0.0000		0.0000		0.0000	
CHC	s				28/0/2		30/0/0		30/0/0
30	p_w				0.0000		0.0000		0.0000
Random	s				24/1/5	28/0/2		29/0/1	
50	p_w				0.0002	0.0000		0.0000	
CHC	s						30/0/0		29/0/1
50	p_w						0.0000		0.0000
Random	s						24/0/6	28/1/1	
100	p_w						0.0003	0.0000	
CHC	s								22/0/7
100	p_w								0.0175
Random	s								17/0/13
200	p_w								0.6884
SVM linear kernel									
Iman–Davenport test p-value: 0.0000									
Random	s		17/0/13	27/0/3		28/0/2		29/0/1	
30	p_w		0.0687	0.0001		0.0000		0.0000	
CHC	s				29/0/1		30/0/0		30/0/0
30	p_w				0.0000		0.0000		0.0000
Random	s				19/2/9	27/1/2		29/0/1	
50	p_w				0.1746	0.0000		0.0000	
CHC	s						23/0/7		25/0/5
50	p_w						0.0021		0.0001
Random	s						10/0/19	25/0/5	
100	p_w						0.0656	0.0002	
CHC	s								26/0/4
100	p_w								0.0001
Random	s								9/0/21
200	p_w								0.0075

shows that for three of the base learners (neural networks, C4.5 and SVM Gaussian) the dense codes were better than both one-vs-all and one-vs-one, but for the remaining three base learners there was no clear tendency. For a SVM, one-vs-one is usually used as the standard binarization method. However, the results show that this is a good idea for linear kernels, where one-vs-one is the best method, but not for Gaussian kernels, for which ECOC codes perform significantly better than both one-vs-one and one-vs-all.

Interestingly enough, if we consider only the three best base learners (neural networks, C4.5 and Gaussian SVMs) we find a clear trend: the ECOC codes outperformed the one-vs-one and one-vs-all methods. Therefore, we can state that when powerful learners are used, ECOC codes are the best choice. The same trend is observed for sparse codes (results not shown for the sake of brevity). One-vs-one is one of the methods more commonly used for class binarization, so it is important that the superior performance of ECOC codes be fully assessed. To further explore the relative behavior of the one-vs-one and ECOC methods, we have compared the differences of both methods for each tested problem. The differences were assessed using a t-test.[4] Table 11 shows a comparison of all problems for one-vs-one, random dense codes of 100 bits (or the longest available) and random sparse codes of 100 bits. The table shows statistical differences between one-vs-one and dense/sparse codes for each problem. This table, together with the results in Table 10, shows a clear pattern of behavior. Powerful base learners, such as neural networks, decision trees and SVMs with a Gaussian kernel, are able to successfully solve the dichotomies posed by the ECOC method, and thus the performance of the ECOC method is better than one-vs-one. On the other hand, less powerful base learners, such as SVMs with a linear kernel, logistic regression and the naive Bayes classifier, more easily solve the simpler dichotomies induced by one-vs-one; thus, their performance in the one-vs-one method

[4] As we are using the 5×2cv setup we can use the 5×2cv F test [47] as well. However, in our experiments this test showed low power and low replicability.

Table 6
Comparison of results for ECOC sparse codes in terms of testing error using a SVM with a Gaussian kernel, logistic regression and a naive Bayes classifier.

		30 bits		50 bits		100 bits		200 bits	
		Random	CHC	Random	CHC	Random	CHC	Random	CHC
SVM Gaussian kernel									
Iman–Davenport test p-value: 0.0000									
Random	s		26/0/4	28/0/2		28/0/2		29/0/1	
30	p_w		0.0002	0.0000		0.0000		0.0000	
CHC	s				26/0/4		29/0/1		28/0/1
30	p_w				0.0000		0.0000		0.0000
Random	s				27/0/3	27/0/3		28/0/2	
50	p_w				0.0001	0.0000		0.0000	
CHC	s						25/0/5		27/0/3
50	p_w						0.0013		0.0000
Random	s						22/0/8	23/1/6	
100	p_w						0.0207	0.0012	
CHC	s								24/1/5
100	p_w								0.0003
Random	s								20/1/9
200	p_w								0.0243
Logistic regression									
Iman–Davenport test p-value: 0.0000									
Random	s		23/0/7	27/0/3		28/0/2		30/0/0	
30	p_w		0.0193	0.0000		0.0000		0.0000	
CHC	s				24/0/6		26/0/4		26/1/3
30	p_w				0.0001		0.0000		0.0000
Random	s				17/0/13	27/0/3		30/0/0	
50	p_w				0.7813	0.0000		0.0000	
CHC	s						28/0/2		27/0/3
50	p_w						0.0000		0.0000
Random	s						13/0/17	27/0/3	
100	p_w						0.2059	0.0000	
CHC	s								26/0/4
100	p_w								0.0000
Random	s								11/0/19
200	p_w								0.0545
Naive Bayes classifier									
Iman–Davenport test p-value: 0.0000									
Random	s		26/0/4	30/0/0		30/0/0		30/0/0	
30	p_w		0.0001	0.0000		0.0000		0.0000	
CHC	s				22/0/8		28/0/2		28/0/2
30	p_w				0.0007		0.0000		0.0000
Random	s				17/0/13	26/0/4		28/0/2	
50	p_w				0.4405	0.0000		0.0000	
CHC	s						27/0/3		30/0/0
50	p_w						0.0000		0.0000
Random	s						12/0/18	26/0/4	
100	p_w						0.0859	0.0001	
CHC	s								25/1/4
100	p_w								0.0000
Random	s								10/0/20
200	p_w								0.0752

is superior to their performance in the ECOC method. This means that the selected class binarization method must be chosen together with the base learner.

Experimental conclusion 7. One-vs-all. The overall results show a worse behavior of one-vs-all with respect to the one-vs-one and ECOC codes.

Previous results. Many previous works have shown these results, but none of them have made a comparison using as many datasets and classifiers as ours. Furthermore, the comparison between one-vs-one and one-vs-all for a SVM with a Gaussian kernel offers an explanation of the discrepancies between Rifkin and Klautau [16] and many other papers.

Experimental conclusion 8. One-vs-one, one-vs-all and ECOC codes. ECOC codes perform better than one-vs-one and one-vs-all when powerful learners are used. When less powerful base learners are used, one-vs-one is able to beat the ECOC method.

Previous results. Allwein et al. [18] compared dense and sparse ECOC codes, one-vs-one and one-vs-all with 8 datasets for a SVM with a polynomial kernel and 13 datasets for ADABOOST with decision stumps as base learners. For SVMs, they found one-vs-all to be worse than the other three. However, they performed no parameter setup. For ADABOOST with decision stumps, they found no differences among the codings. Hsu and Lin [22] compared one-vs-one, one-vs-all and two native multiclass methods for a SVM and 10 datasets using a Gaussian kernel and 6 datasets using a linear kernel. Their overall results showed a worse performance of one-vs-all. However, although parameter setting was performed, the same values were used for all the binary dichotomies induced for a certain dataset. Our experiments showed that this is a suboptimal way of selecting parameters, as the best parameters for each training can differ significantly depending on the induced two-class problem at hand. No previous paper has performed a comparison using six different base learners and compared the conclusions depending on the learner.

Table 7
Comparison of results for ECOC dense and sparse codes in terms of testing error.

Sparse codes		Dense codes					
		Neural net	C4.5	SVM linear	SVM Gaussian	LR	NBC
Random	s	18/0/12	28/0/2	11/0/19	29/0/1	11/0/19	19/0/11
30	p_s	0.3616	0.0000	0.2005	0.0000	0.2005	0.2005
	p_w	0.2536	0.0000	0.0060	0.0000	0.0196	0.1779
CHC	s	15/1/14	26/1/3	4/0/26	24/0/6	4/1/25	11/0/19
30	p_s	1.0000	0.0000	0.0001	0.0014	0.0001	0.2005
	p_w	0.6509	0.0000	0.0000	0.0009	0.0000	0.1470
Random	s	12/1/13	22/0/4	3/0/23	23/0/3	5/0/21	11/0/15
50	p_s	1.0000	0.0005	0.0001	0.0001	0.0025	0.5572
	p_w	0.6567	0.0001	0.0001	0.0001	0.0003	0.1909
CHC	s	8/0/18	22/0/4	2/0/24	18/1/6	6/0/20	12/0/14
50	p_s	0.0755	0.0005	0.0000	0.0227	0.0094	0.8450
	p_w	0.0153	0.0014	0.0000	0.0059	0.0004	0.5506
Random	s	7/1/15	18/0/5	2/0/21	19/0/4	4/0/19	5/0/18
100	p_s	0.1338	0.0106	0.0001	0.0026	0.0026	0.0106
	p_w	0.0614	0.0011	0.0001	0.0007	0.0003	0.0097
CHC	s	3/0/20	15/0/8	2/0/21	19/0/4	3/0/20	7/0/16
100	p_s	0.0005	0.2100	0.0001	0.0026	0.0005	0.0931
	p_w	0.0056	0.4470	0.0001	0.0089	0.0001	0.0163
Random	s	4/0/18	17/0/5	2/0/20	17/0/5	3/0/19	5/0/17
200	p_s	0.0043	0.0169	0.0001	0.0169	0.0009	0.0169
	p_w	0.0007	0.0575	0.0001	0.0074	0.0002	0.0022
CHC	s	4/0/18	12/0/10	2/0/20	15/0/7	2/0/20	5/0/17
200	p_s	0.0043	0.8318	0.0001	0.1338	0.0001	0.0169
	p_w	0.0011	0.9095	0.0001	0.0377	0.0001	0.0041

Table 8
Comparison of random sparse codes of 100 bits using different probabilities of having {−1,0,+1} in the coding matrix.

$P(0) = 1/2$		$P(0) = P(-1) = P(1) = 1/3$					
$P(-1) = P(1) = 1/4$		Neural network	C4.5	SVM linear	SVM Gaussian	LR	NBC
	s	16/0/14	25/0/5	4/1/25	22/0/7	11/0/19	13/0/17
Random	p_s	0.8555	0.0003	0.0001	0.0081	0.2005	0.5847
	p_w	0.7036	0.0001	0.0001	0.0005	0.0350	0.2623
	s	14/1/15	11/0/19	13/1/16	24/0/5	11/0/19	15/0/15
CHC	p_s	1.0000	0.2005	0.7111	0.0005	0.2005	1.0000
	p_w	0.7343	0.3389	0.3441	0.0003	0.0786	0.1109

It may be argued that a comparison between the one-vs-one and ECOC methods is unfair, as for some problems the number of binary classifiers of both methods is quite different. To fully assess the differences between these two methods we carried out the experiments using ECOC dense and sparse codes choosing the bit code length based on the number of classifiers in the one-vs-one method. That is, instead of using a fixed length for the codes, we used $K(K-1)/2$ bits, where K is the number of classes. Table 12 shows a comparison between the one-vs-one and ECOC codes using this setup. The table shows that the general comparison between the ECOC codes and one-vs-one is different, with a certain advantage towards one-vs-one. The reason for this advantage is that, although for some problems the longer codes used are able to improve the performance of the ECOC codes, the shorter codes in other problems have a damaging effect on the performance of the ECOC method.

Table 13 shows a comparison of each dataset. The table shows a different behavior from the one shown in Table 11. The results for the ECOC codes were worse, especially when the number of classifiers was less than 100 bits. Consequently, one-vs-one was better than ECOC, both dense and sparse, for a SVM with a linear kernel, logistic regression and NBC with p-values of the Wilcoxon test all below 0.01, which means a confidence level of 99%. For a neural network one-vs-one was no worse than both ECOC codes, beaten only by C4.5 and a SVM with a Gaussian kernel. If we compare these results with Table 11 we can see a degraded behavior in the ECOC codes.

However, we must take into account that a comparison that limits the number of bits of the code to the number of the one-vs-one learners may be unfair with respect to the ECOC method. For one-vs-one this limit is inherent. However, ECOC method can improve its performance using more learners, which must be considered an advantage of the method.

5.3. The relationship between binary classifier error and coding performance

Several researchers have hypothesized [18] that improving the error-correcting capabilities of ECOC codes may yield more difficult binary problems, harming the performance of the classifier. Others have raised the issue of whether improving the Hamming distance among codewords is actually increases the independence of the binary classifiers [26].

To assess this hypothesis, we tested the accuracy of the binary classifiers for the coding methods we are comparing. Table 14 shows a comparison of the different methods for the ECOC dense and sparse codes and for the one-vs-one and one-vs-all methods. Fig. 2 shows the binary testing error results for the one-vs-all and one-vs-one methods and all the base learners. Figs. 3 shows the binary testing error results for the ECOC dense and sparse codes for each base learner.

Regarding the ECOC codes, our results strongly support the idea that error-correcting codes yield to more difficult dichotomies than random codes, as the accuracy of the binary classifiers was worse

Table 9
Comparison of results for random dense codes (top) and sparse codes (bottom) in terms of testing error using codewords of 100 bits.

		Neural net	C4.5	SVM-l	SVM-g	LR	NBC
	s		17/0/13	6/0/24	16/0/14	3/0/27	5/0/25
Neural net	p_s		0.5847	0.0014	0.8555	0.0000	0.0003
	p_w		0.3493	0.0001	0.7971	0.0000	0.0000
	s			9/0/21	15/0/15	7/0/23	2/0/28
C4.5	p_s			0.0428	1.0000	0.0052	0.0000
	p_w			0.0004	0.6435	0.0001	0.0000
	s				23/0/7	7/0/23	10/0/20
SVM-l	p_s				0.0052	0.0052	0.0987
	p_w				0.0068	0.0009	0.0196
	s					7/0/23	4/0/26
SVN-g	p_s					0.0052	0.0001
	p_w					0.0015	0.0000
	s						11/0/19
LR	p_s						0.2005
	p_w						0.1779
	s		12/0/18	11/0/19	14/0/16	7/0/23	4/0/26
Neural net	p_s		0.3616	0.2005	0.8555	0.0052	0.0001
	p_w		0.5716	0.0387	0.1986	0.0005	0.0000
	s			13/0/17	14/0/16	9/0/21	1/0/29
C4.5	p_s			0.5847	0.8555	0.0428	0.0000
	p_w			0.1306	0.2712	0.0032	0.0000
	s				16/0/14	6/1/23	9/0/21
SVM-l	p_s				0.8555	0.0023	0.0428
	p_w				0.7499	0.0019	0.0014
	s					12/0/18	6/0/24
SVN-g	p_s					0.3616	0.0014
	p_w					0.1846	0.0005
	s						8/0/22
LR	p_s						0.0161
	p_w						0.0148

Table 10
Comparison of results for one-vs-one, one-vs-all and ECOC dense codes methods in terms of testing error.

Hhline––		Neural net		C4.5		SVM linear	
		One-vs-all	ECOC	One-vs-all	ECOC	One-vs-all	ECOC
	s	2/0/42	22/0/8	7/0/37	27/0/3	6/0/38	4/0/26
One-vs-one	p_s	0.0000	0.0161	0.0000	0.0000	0.0000	0.0001
	p_w	0.0000	0.0368	0.0000	0.0000	0.0000	0.0000
	s		30/0/0		28/1/1		12/1/17
One-vs-all	p_s		0.0000		0.0000		0.4583
	p_w		0.0000		0.0000		0.5372
		SVM Gaussian		Logistic regression		Naive Bayes	
	s	31/1/12	29/0/1	8/0/36	6/0/24	17/0/27	9/1/20
One-vs-one	p_s	0.0054	0.0000	0.0000	0.0014	0.1742	0.0614
	p_w	0.0123	0.0000	0.0000	0.0012	0.0486	0.0152
	s		28/0/2		11/0/19		14/0/16
One-vs-all	p_s		0.0000		0.2005		0.8555
	p_w		0.0000		0.5577		0.5304

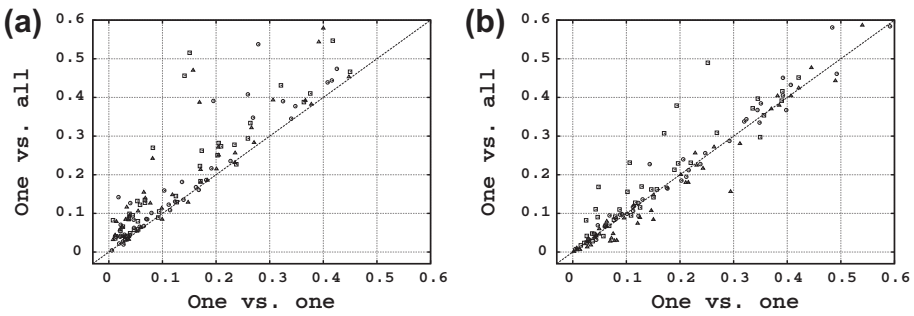

Fig. 1. Testing error results for one-vs-all and one-vs-one for (a) a neural network (circles), a C4.5 tree (triangles), and a SVM (squares) with a linear kernel as base learners; and for (b) a SVM with a Gaussian kernel (circles), logistic regression (triangles), and a naive Bayes classifier (squares) as base learners.

Table 11
Comparison of one-vs-one and ECOC codes, both dense and sparse, for each dataset. Statistically significant differences in favor of ECOC codes are marked with × and in favor of the one-vs-one method with ✓. No significant differences are shown with —.

Dataset	Neural networks		C4.5		SVM linear		SVM Gaussian		Linear regression		Naive Bayes	
	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse
Abalone	–	–	—	—	✓	✓	×	—	✓	✓	—	—
Arrhythmia	×	×	×	×	✓	—	×	×	—	—	—	—
Audiology	–	–	×	×	✓	—	×	×	×	×	—	—
Autos	–	–	—	×	✓	—	×	×	✓	—	—	—
Dermatology	–	–	—	×	—	—	×	×	—	—	✓	—
Ecoli	×	×	—	—	—	—	×	×	—	—	—	—
glass	–	×	—	—	—	—	×	—	—	—	—	—
Isolet	–	×	×	×	✓	✓	×	×	✓	✓	✓	✓
Krkopt	✓	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓
Led24	–	–	×	×	—	—	×	×	✓	—	—	—
Letter	✓	✓	×	×	✓	✓	×	×	✓	✓	✓	✓
Lrs —	–	–	×	×	✓	—	×	—	✓	—	×	×
Mfeat-fac	×	×	×	×	—	—	×	×	×	×	—	—
Mfeat-fou	×	×	×	×	✓	—	×	×	✓	—	—	—
Mfeat-kar	×	×	×	×	✓	—	×	×	✓	×	—	—
Mfeat-mor	–	–	×	×	✓	✓	×	×	✓	✓	—	—
Mfeat-pix	×	×	×	×	✓	✓	×	×	✓	×	✓	✓
Mfeat-zer	×	×	×	×	✓	—	×	—	✓	×	—	✓
Optdigits	×	×	×	×	✓	✓	×	×	✓	—	✓	✓
Pendigits	–	×	×	×	✓	✓	×	×	✓	✓	✓	✓
Primary-tumor	×	×	—	—	×	—	×	×	×	—	—	×
Satimage	–	×	×	×	✓	✓	×	×	—	—	×	×
Segment	–	–	×	×	✓	✓	×	×	—	—	✓	✓
Shuttle	–	×	—	×	✓	✓	×	×	✓	✓	✓	—
Soybean	–	–	×	—	—	—	×	×	×	—	✓	✓
Texture	✓	–	×	×	✓	×	×	×	—	×	✓	✓
Vowel	✓	–	×	×	✓	✓	×	×	✓	✓	—	—
Yeast	×	×	×	×	✓	✓	×	×	✓	—	×	×
Zip —	–	×	×	×	✓	—	—	—	✓	—	—	×
Zoo —	–	×	—	—	—	—	×	×	×	—	—	—
Win/loss	10/4	17/2	22/0	24/0	1/22	1/13	28/1	24/1	5/18	6/8	3/11	5/10

Table 12
Comparison of one-vs-one and ECOC dense and sparse codes using codes of $K(K-1)/2$ bits.

		One-vs-one					
		Neural network	C4.5	SVM linear	SVM Gaussian	LR	NBC
Dense codes	s	15/1/14	3/0/27	28/0/2	1/0/29	24/0/6	22/1/7
	p_s	1.0000	0.0000	0.0000	0.0000	0.0014	0.0081
	p_w	0.6216	0.0000	0.0000	0.0000	0.0002	0.0004
Sparse codes	s	16/0/14	7/0/23	27/0/3	5/0/25	24/0/6	23/0/7
	p_s	0.8555	0.0052	0.0000	0.0003	0.0014	0.0052
	p_w	0.6884	0.0093	0.0000	0.0001	0.0001	0.0004

for the CHC codes than for the random codes. The differences were statistically significant for all of the base learners, with all p-values of the Wilcoxon test below 0.01. However, in general, the differences for a given problem were not large. Nevertheless, this worse performance of binary classifiers undermines the possible advantages of CHC codes. This may be one of the reasons why CHC codes are not able to improve upon the performance of random codes.

It is also usually assumed that the two-class problems that we must solve in the one-vs-all method are harder to learn than the problems induced by the one-vs-one method. A comparison between the binary errors of the one-vs-one and one-vs-all learners must be made with caution, as the problems addressed by these methods are different. Therefore, we must bear in mind that we are comparing the difficulty of the problems and not the quality of the classifiers. Nevertheless, the assumption of easier problems for one-vs-one was not fully corroborated by experiments. Only for neural networks were the binary errors for one-vs-one significantly smaller than for one-vs-all. For C4.5 and the Gaussian SVMs, the one-vs-all binary errors were even significantly smaller than the binary errors of one-vs-one. Thus, as a general rule the accuracy achieved by the binary classifiers in the one-vs-all method is usually as good as the accuracy of the classifiers in the one-vs-one method. We must take into account that, although separating two classes may be easier than separating a class from all the remaining classes, the number of available instances for the former problem is much less than the number of available instances for the latter. Consequently, this last problem is more susceptible to over-fitting.[5]

It is noticeable that for some problems, namely abalone, arrhythmia, audiology, and primary-tumor, the minimum testing accuracy of the binary classifiers for one-vs-one method was very low. A closer look at the results showed that this problem appeared in datasets with many classes. For some pairs, the number of instances belonging to any of the two classes was very low, yielding to poorly trained binary classifiers. These classifiers might also have a harmful effect on the overall accuracy of the classifier. This problem did not arise in one-vs-all methods, as all binary classifiers were trained with all the data.

[5] In fact, the training accuracy for the binary classifiers was always better for the one-vs-one method.

Table 13
Comparison of one-vs-one and ECOC codes, both dense and sparse, with codewords of $K(K-1)/2$ bits for each dataset. Statistically significant differences in favor of ECOC codes are marked with × and in favor of one-vs-one method with ✓. No significant differences are shown with a —.

Dataset	Neural networks		C4.5		SVM linear		SVM Gaussian		Linear regression		Naive Bayes	
	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse
Abalone	×	×	×	×	✓	✓	×	✓	✓	✓	×	×
Arrhythmia	×	×	×	×	✓	✓	×	—	—	×	—	—
Audiology	—	—	×	×	—	—	×	×	×	×	—	—
Autos	—	✓	—	—	✓	✓	—	—	✓	✓	—	—
Dermatology	—	—	—	—	—	✓	×	×	—	—	✓	✓
Ecoli	—	—	—	—	—	—	×	—	—	—	—	✓
Glass	—	—	—	—	✓	—	×	—	✓	✓	—	—
Isolet	×	×	×	×	✓	✓	×	×	✓	✓	✓	✓
Krkopt	✓	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓
Led24	—	—	—	—	✓	—	×	×	✓	✓	✓	✓
Letter	✓	×	×	×	✓	✓	×	×	✓	✓	✓	✓
Lrs —	—	—	×	×	✓	—	×	×	✓	—	×	—
Mfeat-fac	—	—	×	×	✓	✓	×	×	×	×	—	—
Mfeat-fou	—	—	×	×	✓	✓	×	×	✓	✓	—	✓
Mfeat-kar	×	—	×	×	✓	✓	×	×	✓	—	✓	✓
Mfeat-mor	—	—	×	—	✓	✓	×	×	✓	✓	—	—
Mfeat-pix	×	—	×	×	✓	✓	×	×	✓	×	✓	✓
Mfeat-zer	—	—	×	×	✓	—	×	×	✓	✓	✓	✓
Optdigits	—	—	×	×	✓	✓	×	×	✓	✓	✓	✓
Pendigits	—	×	×	—	✓	✓	×	—	✓	✓	✓	✓
primary-tumor	×	×	×	—	×	—	×	×	×	×	—	×
Satimage	✓	—	×	—	✓	✓	×	—	✓	✓	—	✓
Segment	—	—	×	×	✓	✓	—	✓	✓	—	✓	✓
Shuttle	—	—	—	—	—	—	×	—	✓	✓	✓	✓
Soybean	—	×	×	×	—	—	×	×	×	×	✓	✓
Texture	✓	✓	×	×	✓	✓	×	×	✓	—	✓	✓
Vowel	✓	✓	×	×	✓	✓	×	×	✓	✓	—	✓
Yeast	×	×	×	×	✓	✓	×	×	✓	✓	×	×
Zip —	✓	—	×	×	✓	✓	—	—	✓	✓	✓	—
Zoo —	—	—	—	—	—	—	×	×	—	—	—	—
Win/loss	7/6	8/4	23/0	20/0	0/24	0/20	26/1	19/3	4/22	3/21	3/15	3/18

Table 14
Comparison of the binary classifiers testing accuracy for ECOC dense and sparse codes, one-vs-one and one-vs-all.

		Neural network	C4.5	SVM linear	SVM Gaussian	LR	NBC
Dense codes (CHC)							
	s	8/0/22	0/1/29	2/0/28	4/0/26	0/0/30	0/0/30
Random	p_s	0.0161	0.0000	0.0000	0.0001	0.0000	0.0000
	p_w	0.0001	0.0000	0.0000	0.0001	0.0000	0.0000
Sparse codes (CHC)							
	s	3/0/27	4/0/26	4/0/26	7/0/23	4/0/26	2/0/28
Random	p_s	0.0000	0.0001	0.0001	0.0052	0.0001	0.0000
	p_w	0.0000	0.0000	0.0000	0.0017	0.0000	0.0000
One-vs-all							
	s	9/0/35	24/0/20	14/0/30	35/0/9	17/0/27	22/0/22
One-vs-one	p_s	0.0001	0.6516	0.0226	0.0001	0.1742	1.0000
	p_w	0.0001	0.0441	0.2882	0.0001	0.3327	0.5210

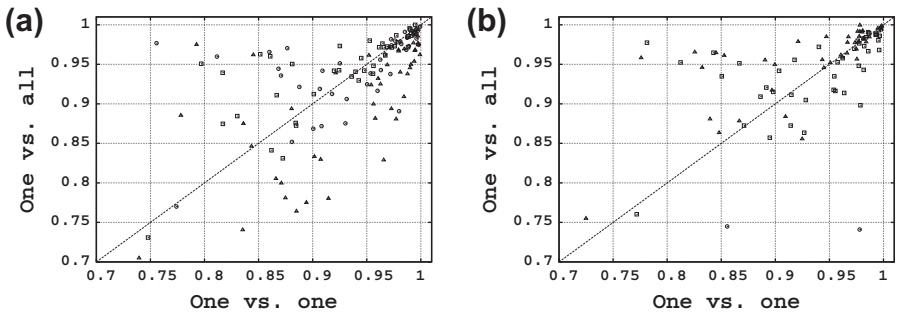

Fig. 2. Binary testing accuracy for one-vs-all and one-vs-one for (a) a neural network (circles), a C4.5 tree (triangles), and a SVM (squares) with a linear kernel as base learners; and for (b) a SVM with a Gaussian kernel (circles), logistic regression (triangles), and a naive Bayes classifier (squares) as base learners.

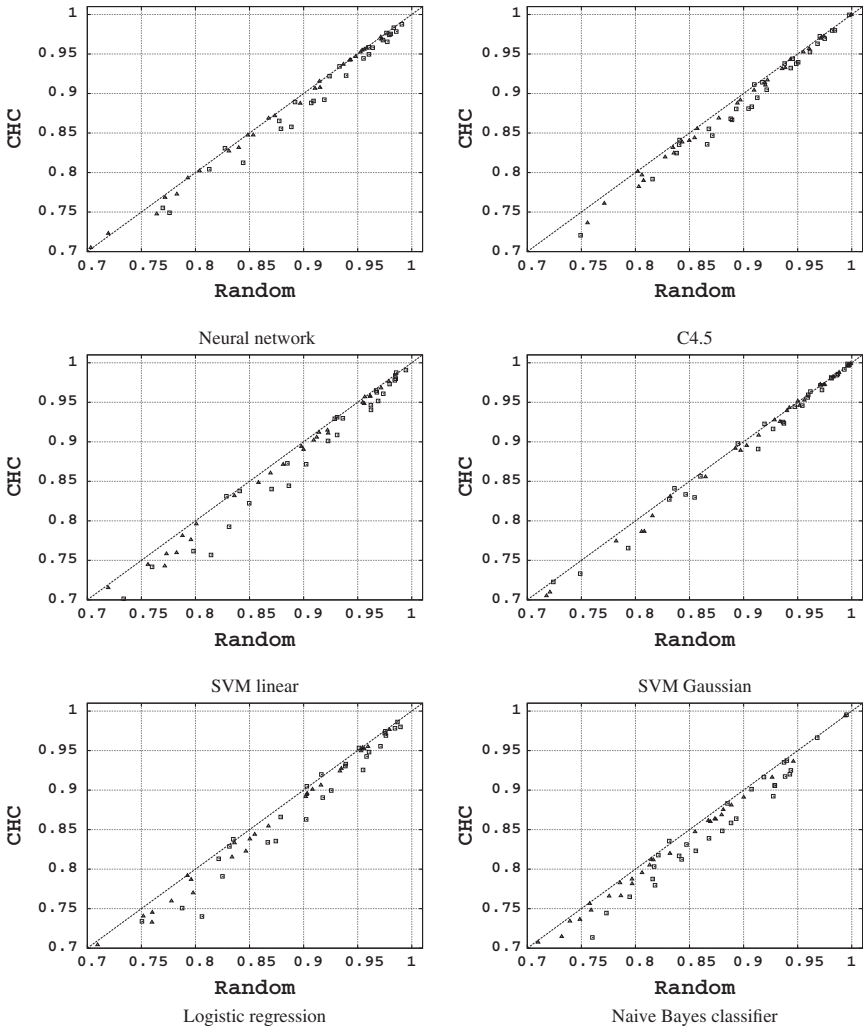

Fig. 3. Binary testing accuracy for ECOC dense codes (triangles) and sparse codes (squares) using the six different base learners.

Experimental conclusion 9. Binary classifier testing error. The binary testing error of the CHC codes is worse than for random codes for both dense and sparse codes. The comparison between one-vs-one and one-vs-all depends on the base learner used.

Previous results No previous work have carried out this study. However, these results corroborate previous methods based on performing the coding design together with the training of the classifier [48] to avoid introducing more difficult dichotomies.

5.4. Independence of classifiers

In the previous section we showed that the dichotomies induced by CHC codes are more difficult than the dichotomies induced by random codes. It is assumed that codes designed by their error-correcting capabilities should improve the independence of the errors between the classifiers. Consequently, their failure to improve the performance of random codes is attributed to the more difficult dichotomies induced. However, whether the obtained classifiers are more independent is not an established fact. In this section we study whether this assumption of more independent errors is justified.

We measured the independence of the classifiers using Yule's Q statistic. The Yule's Q statistic [49] for two classifiers, D_i and D_k, is given by:

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}, \tag{8}$$

where N^{ab} is given by:

	D_k correct (1)	D_k wrong (0)
D_i correct (1)	N^{11}	N^{10}
D_i wrong (0)	N^{01}	N^{00}

We selected this measure, because it proved to have the best results in a recent paper comparing ten different diversity measures [50]. Classifiers that recognize the same patterns will have positive values of Q, and classifiers that tend to make mistakes in different patterns will have negative values of Q. For independent classifiers the expectation of Q is 0. For a set of L classifiers we use an average value Q_{av}:

$$Q_{av} = \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{k=i+1}^{L} Q_{i,k}. \tag{9}$$

Table 15 shows a comparison for the six base learners. For both codes, our experiments corroborated that error-correcting codes were able to improve the independence of the base learners for all cases, except for dense codes using a neural network. All of the remaining p-values are below 0.05, and the win/draw/loss records show a marked difference.

Table 15
Comparison of average Q values for the six base learners.

		Neural network	C4.5	SVM linear	SVM Gaussian	LR	NBC
Dense codes (CHC)							
	s	17/0/13	25/1/4	27/0/3	17/4/9	28/0/2	39/0/1
Random	p_s	0.5847	0.0001	0.0000	0.1686	0.0000	0.0000
	p_w	0.3389	0.0000	0.0000	0.0157	0.0000	0.0000
Sparse codes (CHC)							
	s	29/0/1	28/0/2	29/0/1	30/0/0	29/0/1	29/0/1
Random	p_s	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	p_w	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000

Furthermore, the sparse codes always achieved smaller absolute values of Q than the dense codes. The reason for this effect may be found in the differences between both types of codes. For dense codes, all the binary classifiers are trained using all the data, so although the dichotomies are different, it is more difficult to obtain independent classifiers as all classifiers are trained using the same data. On the other hand, sparse codes disregard the instances of classes that have a 0 in the corresponding column representing the dichotomy. The CHC algorithm enforces column separation, which means that the columns have less overlap. Thus, the binary classifiers induced by the CHC matrices are trained using datasets that have less overlap, and they can be more independent. Unfortunately, although the independence is higher, the binary testing error is also higher (see Section 5.3), which undermines the improvement that might be obtained from the more diverse binary classifiers. In this way, the behavior of ECOC can be improved by combining binary classifiers that are independent and accurate [48]. As we have stated, the combination of weak learners can take advantage of the independence of the errors they make, whereas combining powerful learners is less profitable due to their more correlated errors. The obvious reason for this effect is that as the classifiers are more accurate there is less room for disagreement.

Experimental conclusion 10. Independence of binary errors. For dense codes and sparse codes, the independence of classifiers for CHC codes is higher than for random codes.
Previous results. Masulli and Valentini [51] compared the independence of ECOC codes using neural networks to learn the dichotomies and a single neural network to learn the coding.

5.5. Comparison with native multiclass methods

As we have stated some algorithms are specifically designed for two-class problems. However, there are other algorithms that can handle multiclass problems. For these algorithms, the obvious question is whether the multiclass approaches studied in this paper are better than the native multiclass approach. We compare these with one-vs-one, one-vs-all, and ECOC dense and sparse random codes of 100 bits or the longest available.

Of the base learners studied in the previous sections, three have widely used native multiclass methods. In this section we compare the performance of the standard multiclass methods of neural networks and C4.5 against the class binarization methods. NBC was not used as the native multiclass method achieved poor results. Table 16 shows a comparison of these methods for C4.5 and neural networks. For C4.5, we used the multiclass method provided with the program [52]. We used a neural network with 25 hidden nodes and the standard 1-out-of-N codification of the classes. The case for SVMs was different, as there are different multiclass methods [53,16] available, but none of them is used as a standard for multiclass SVM. However, for the sake of completeness, we also compared output coding methods with the SVM multiclass method described in [53]. The parameters of the multiclass SVM were adjusted using the same procedure described for binary SVMs.

Table 16
Comparison of results for native and binarization methods using a neural network, a C4.5 tree an a SVM as base learners.

		Neural net			
		One-vs-one	One-vs-all	ECOC dense	ECOC sparse
	s	27/0/17	3/0/41	24/0/6	24/0/6
Native	p_s	0.1742	0.0000	0.0014	0.0014
	p_w	0.0832	0.0000	0.0013	0.0002
		C4.5			
	s	33/0/11	8/0/36	23/0/7	24/0/6
Native	p_s	0.0013	0.0000	0.0052	0.0014
	p_w	0.0026	0.0000	0.0015	0.0002
		Linear SVM			
	s	26/0/18	29/0/15	14/0/16	19/0/11
Native	p_s	0.2912	0.0488	0.8555	0.2005
	p_w	0.6618	0.7372	0.0975	0.0975
		Gaussian SVM			
	s	19/0/25	26/0/18	9/0/21	18/0/12
Native	p_s	0.4514	0.2912	0.0428	0.3616
	p_w	0.7372	0.7372	0.2882	0.2882

For the neural networks and C4.5 learners we found a similar behavior. The one-vs-one method performed better than the native method, and the one-vs-all method was outperformed by the standard multiclass method. However, for the neural network, the differences between one-vs-one and the native method were not significant at a 95% confidence level, with a p-value of 0.0832. The ECOC codes, both dense and sparse, were able to significantly improve the performance of the standard multiclass methods for C4.5 and the neural networks with p-values below 0.05. Nevertheless, we must take into account that the class binarization methods are more complex, and they need more time for training and recall. The case for SVMs was different. Using a linear kernel, the class binarization methods performed slightly better than the native multiclass method, but the differences were not significant at a 95% confidence level. Using a Gaussian kernel the differences were even less marked. However, if we consider individual results, we can observe an interesting effect. Although the multiclass method had a good performance, it had the clear disadvantage of being less stable, with testing errors that, depending on the problem could be quite good or quite bad.

Experimental conclusion 11. Comparison with native multiclass methods. ECOC codes, both dense and sparse, and one-vs-one perform better than native multiclass methods for C4.5 and neural networks, whereas one-vs-all is consistently beaten by native multiclass methods. For SVM, using linear or Gaussian kernels, the differences are not significant overall, although they can be marked for individual problems.
Previous results. Hsu and Lin [22] performed a comparison of one-vs-one, one-vs-all, DDAG, and two multiclass native

methods for SVM. Although they concluded that one-vs-all is inferior to one-vs-one, the experiments were not so conclusive, as only 10 datasets were considered, and only two of them had more than seven classes. Furthermore, for seven problems, all of the methods resulted in an error below 5%, leaving little room for significant improvement.

5.6. Problems with many classes

It is desirable for any study to identify the kind of problems that are most appropriate for the compared methods. However, such identification is quite difficult and is not available for most of the classification methods; however, such identification is still worth trying. In this section we compare the methods studied in the specific scenario of problems with many classes. The aim is to test whether any of the methods are more suitable for these kinds of problems, which are common in data mining applications. We chose random codewords of 100 bits to represent ECOC codes. For these experiments, we considered the three best base learners: neural networks, C4.5 and a SVM with a Gaussian kernel.

We considered problems with 10 or more classes. Table 17 shows a comparison of the three methods. Again, the table shows that the behavior of the three methods depends on the base learner used. If we compare this table with the table comparing the methods for all the datasets, no large differences are found. The relative behavior of the methods is the same for problems with many classes as problems with fewer classes. The only difference is in one-vs-one and one-vs-all for SVM, where the performance of one-vs-all was worse than when all problems were considered.

Table 17
Comparison of results for problems with more than 10 classes.

		Neural net		
		One-vs-all	Dense	Sparse
One-vs-one	s	0/0/22	19/0/6	18/0/4
	p_s	0.0000	0.0525	0.0043
	p_w	0.0000	0.1886	0.0108
One-vs-all	s	22/0/0	22/0/0	
	p_s	0.0000	0.0000	
	p_w		0.0000	0.0000
Dense	s			15/1/6
	p_s			0.0784
	p_w			0.0321
	C4.5			
One-vs-one	s	2/0/20	21/0/1	22/0/0
	p_s	0.0001	0.0000	0.0000
	p_w	0.0001	0.0000	0.0000
One-vs-all	s		22/0/0	22/0/0
	p_s		0.0000	0.0000
	p_w		0.0000	0.0000
Dense	s			5/0/17
	p_s			0.0169
	p_w			0.0017
	SVM Gaussian			
One-vs-one	s	2/0/20	21/0/1	20/0/2
	p_s	0.0001	0.0000	0.0001
	p_w	0.0001	0.0001	0.0001
One-vs-all	s		20/0/2	17/0/5
	p_s		0.0001	0.0169
	p_w		0.0003	0.0067
Dense	s			4/0/18
	p_s			0.0043
	p_w			0.0012

Experimental conclusion 12. Problems with many classes. On average, the relative performance of the methods is the same when only problems with many classes are considered.

Previous results. No previous work has studied separately the performance of the different methods in problems with many classes.

5.7. Effect of noise

One of the most interesting aspects of any study of classifier performance is the effect class label noise has on the performance of the method studied. It is highly likely that for any real application we would have some instances that are mislabeled. The effect of these instances on the performance of the classifier must be known. In this section we study the effect of artificially added noise to class labels to evaluate how the performance of each method is deteriorated by this noise.

To add noise to the class labels we followed the method in [54]. To add classification noise at a rate r, we chose a fraction r of the instances and changed their class labels to be incorrect choosing uniformly from the set of incorrect labels. We chose all the datasets and three rates of noise, 5%, 10%, and 20%. With these three levels

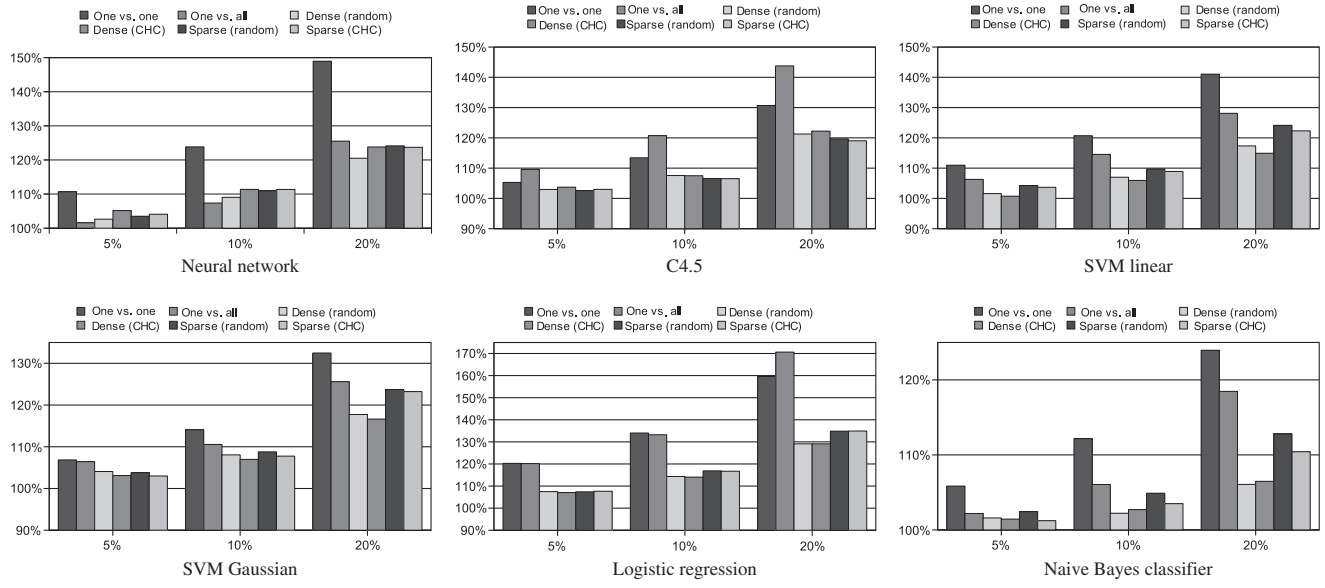

Fig. 4. Relative increment of testing error for noise levels of 5%, 10% and 20%. The increment with respect to the original datasets without artificial noise is shown.

of noise we performed the experiments using the previous methods and the six different base learners.

The behavior of the methods is shown in Fig. 4. In the figure we show the ratio of increment in the average error with respect to the same method applied to original datasets. In this case we had a clear pattern. The ECOC codes, both dense and sparse, were more robust in the presence of noise than one-vs-one and one-vs-all. Furthermore, the dense codes were less affected by noise than the sparse codes, with the exception of C4.5. The reason for this effect may be that dense codes always use all of the instances for training the classifiers, thus reducing the impact of the noisy labels. One-vs-all was more robust than one-vs-one for four of the six base learners. As a general rule, one-vs-one is the most affected by noise. As one-vs-one uses less instances to train the learners, these learners are more affected by the presence of noisy instances.

Experimental conclusion 13. Noise effect. ECOC codes, both dense and sparse, are more robust to noise presence. One-vs-one is the method most affected by noise presence.
Previous results. No previous work has carried out a similar comparison.

6. Conclusions

We have made a study of the different aspects that affect the process of fusion of the different binary classifiers that made a multiclass classifier based on output coding. Thorough comparisons were made between the different methods to solve multiclass problems using only binary classifiers. The experiments show that several aspects of these methods are usually taken for granted without adequate experimental corroboration. These issues have been tested on a benchmark that includes a fairly large number of problems.

Furthermore, the study was not limited to one base learner. Instead, we have used six of the most widely used classification algorithms. One of the most interesting conclusions of our work is that the behavior of binarization methods, an especially one-vs-one, varies depending on the base learner. However, the ECOC codes and one-vs-all showed a more stable behavior regardless of the binary learner. Thus, in a practical application both decisions, the base learner and the binarization method, must be considered together.

We summarize the results of our experiments as follows:

- The performance of ECOC codes improves as longer codewords are added, but the improvement is less dramatic for longer codes. No significant differences between random codes and codes designed specifically for their error-correcting capabilities were found. This behavior was observed using both dense and sparse codes.
- Dense and sparse codes exhibit the same behavior in the absence of noise. However, when noise is added, dense codes are slightly more robust than sparse codes. Furthermore, as a general rule, one-vs-one is the method most affected by noise, which degrades its performance considerably.
- Similar results were obtained with the three most powerful learners used: neural networks, decision trees and SVMs with a Gaussian kernel. As a general rule we cannot say that any of these three learners is better than the other two. However, for a given problem significant differences were found.
- For neural networks and C4.5, standard multiclass algorithms were able to improve the performance of one-vs-all, or at least match the performance of one-vs-one. However, the ECOC codes generally performed better than native multiclass methods.
- Binary testing error deteriorated when the codewords were designed by their error-correcting capabilities, which specially affects weaker learners.
- The independence of classifiers was improved by designing the codes for their error-correcting capabilities, especially sparse codes.

Although recommending a method for any given problem is always risky, it may be concluded from our study that, as a general rule, ECOC codes are the best choice when powerful learners are used, such as decision trees, neural networks or SVMs with a Gaussian kernel. The one-vs-one is the best choice when weaker learners are used. In fact, random ECOC codes can be used without resorting to any complex optimization algorithm for improving error-correcting capabilities. Dense and sparse codes can be used with the same expected performance. Dense codes have the advantage of being more robust to noise, and sparse codes have the advantage of a shorter training time. Although long ECOC codes are more computationally expensive than the one-vs-one and one-vs-all methods, we believe it pays to have added complexity for improved performance. Furthermore, we must take into account that the performance of each method is linked to the base learner used, so both decisions must be taken together.

Acknowledgement

This work was supported in part by the Project TIN2008-03151 of the Spanish Ministry of Education and Science.

References

[1] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: D. Haussler (Ed.), Proceedings of the 5th Annual ACM Workshop on COLT, 1992, pp. 144–152.
[2] E.M. Dos Santos, R. Sabourin, P. Maupin, Overfitting cautious selection of classifiers ensembles with genetic algorithms, Information Fusion 10 (2009) 150–162.
[3] T. Windeatt, R. Ghaderi, Binary labelling and decision-level fusion, Information Fusion 2 (2001) 103–112.
[4] T. Windeatt, Diversity measures for multiple classifier system analysis and design, Information Fusion 6 (2005) 21–36.
[5] Z. He, X. Xu, S. Deng, A cluster ensemble method for clustering categorical data, Information Fusion 6 (2005) 143–151.
[6] P. Clark, R. Boswell, Rule induction with CN2: some recent improvements, in: Proceedings of the 5th European Working Session on Learning (EWSL-91), Springer-Verlag, Porto, Portugal, 1991, pp. 151–163.
[7] R. Anand, K.G. Mehrotra, C.K. Mohan, S. Ranka, Efficient classification for multiclass problems using modular neural networks, IEEE Transactions on Neural Networks 6 (1995) 117–124.
[8] S. Knerr, L. Personnaz, G. Dreyfus, Single-layer learning revisited: a stepwise procedure for building and training a neural network, in: J. Fogelman (Ed.), Neurocomputing: Algorithms, Architectures and Applications, Springer-Verlag, New York, 1990.
[9] T. Hastie, R. Tibshirani, Classification by pairwise coupling, The Annals of Statistics 26 (2) (1998) 451–471.
[10] J.C. Platt, N. Christiani, J. Shawe-Taylor, Large margin DAGs for multiclass classification, in: S.A. Solla, T.K. Leen, K.-R. Müller (Eds.), Proceedings of Neural Information Processing Systems, NIPS'99, MIT Press, 2000, pp. 547–553.
[11] T.-F. Wu, C.-J. Lin, R.C. Weng, Probability estimates for multi-class classification by pairwise coupling, Journal of Machine Learning Research 5 (2004) 975–1005.
[12] N. García-Pedrajas, D. Ortiz-Boyer, Improving multiclass pattern recognition by the combination of two strategies, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (6) (2006) 1001–1006.
[13] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, Journal of Artificial Intelligence Research 2 (1995) 263–286.
[14] T. Windeatt, R. Ghaderi, Coding and decoding strategies for multi-class learning problems, Information Fusion 4 (2003) 11–21.
[15] E.B. Kong, T.G. Dietterich, Why error-correcting output coding works with decision trees, Technical Report, Department of Computer Science, Oregon State University, Corvallis, OR, 1995.
[16] R. Rifkin, A. Klautau, In Defense of One-Vs-All Classification, Journal of Machine Learning Research 5 (2004) 101–141.
[17] M. Moreira, E. Mayoraz, Improved pairwise coupling classification with correcting classifiers, in: Proceedings of the 10th European Conference on Machine Learning, vol. 1398, Lecture Notes in Computer Science, Springer, Chemnitz, Germany, 1998, pp. 160–171, doi:10.1007/BFb0026686.

[18] E.L. Allwein, R.E. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, Journal of Machine Learning Research 1 (2000) 113–141.
[19] J. Fürkranz, Round robin classification, Journal of Machine Learning Research 2 (2002) 721–747.
[20] W. Utschick, Error-correcting classification based on neural networks, Ph.D. thesis, Technische Universitat Munchen, 1998.
[21] E.B. Kong, T.G. Dietterich, Error-correcting output coding corrects bias and variance, in: A. Prieditis, J.F. Lemmer (Eds.), Machine Learning: Proceedings of the Twelfth International Conference, Elsevier Science Publishers, 1995, pp. 275–283.
[22] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, TNN 13 (2) (2002) 415–425.
[23] A. Passerini, M. Pontil, P. Frasconi, New results on error correcting output codes of kernel machines, IEEE Transactions on Neural Networks 15 (1) (2004) 45–54.
[24] J.D.M. Rennie, R. Rifkin, Improving multiclass text classification with the support vector machine, Technical Report 2001-026, MIT, 2001.
[25] F. Masulli, G. Valentini, Effectiveness of error correcting output codes in multiclass learning problems, in: J. Kittler, F. Roli (Eds.), Multiple Classifier Systems, MCS2000, Lecture Notes in Computer Science, Springer, Cagliary, Italy, 2000, pp. 107–116.
[26] F. Masulli, G. Valentini, Effectiveness of error correcting output coding methods in ensemble and monolithic learning machines, Patterns Analysis and Applications 6 (2003) 285–300.
[27] R.E. Shapire, Using output codes to boost multiclass learning problems, in: D.H. Fisher (Ed.), Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 313–321.
[28] G.M. James, T. Hastie, The error coding method and PICT'S, Computational and Graphical Statistics 7 (1998) 377–387.
[29] O. Pujol, P. Radeva, J. Vitriá, Discriminant ECOC: a heuristic method for application dependent design of error correcting output codes, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (6) (2006) 1007–1012.
[30] O. Pujol, S. Escalera, P. Radeva, An incremental node embedding technique for error correcting output codes, Pattern Recognition 41 (2008) 713–725, doi:10.1016/j.patcog.2007.04.008.
[31] S. Escalera, D.M.J. Tax, O. Pujol, P. Radeva, R.P.W. Duin, Subclass problem-dependent design for error-correcting output codes, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (6) (2008) 1041–1054, doi:10.1109/TPAMI.2008.38.
[32] I. Melvin, E. Ie, J. Weston, W.S. Noble, C. Leslie, Multi-class protein classification using adaptive codes, Journal of Machine Learning Research 8 (2007) 1557–1581.
[33] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Computation 10 (7) (1998) 1895–1923.
[34] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–83.
[35] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[36] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Annals of Mathematicals Statistics 11 (1940) 86–92.
[37] R.L. Iman, J.M. Davenport, Approximations of the critical regions of the Friedman statistic, Communications in Statistics 6 (1980) 571–595.
[38] Y. Yang, G.I. Webb, A comparative study of discretization methods for Naive-Bayes classifiers, in: Proceedings of the 2002 Pacific Rim Knowledge Acquisition Workshop (PKAW 2002), Tokyo, Japan, 2002, pp. 159–173.
[39] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, Chambéry, France, 1993, pp. 1022–1029.
[40] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines. <http://www.csie.ntu.edu.tw/cjlin/libsvm>, 2001.
[41] R.C. Bose, D.K. Ray-Chaudhuri, On a class of error-correcting binary group codes, Information and Control 3 (1960) 68–79.
[42] L.I. Kuncheva, Using diversity measures for generating error-correcting output codes, Pattern Recognition Letters 26 (2005) 83–90.
[43] N. Hatami, S. Seyedtabaii, Error correcting output codes using genetic algorithm-based decoding, in: Proceedings of the Fourth International Conference on Networked Computing and Advanced Information Management, vol. 1, IEEE Computer Society, Los Alamitos, CA, USA, 2008, pp. 391–396, doi:10.1109/NCM.2008.260.
[44] D. Ortiz-Boyer, C. Hervás-Martínez, N. García-Pedrajas, CIXL2: A crossover operator for evolutionary algorithms based on population features, Journal of Artificial Intelligence Research 24 (2005) 33–80.
[45] L.J. Eshelman, The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, Morgan Kauffman, San Mateo, CA, 1990.
[46] D.H. Wolpert, W.G. Macready, No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.
[47] E. Alpaydin, Combined 5×2cv F test for comparing supervised classification learning algorithms, Neural Computation 11 (1999) 1885–1892.
[48] N. García-Pedrajas, C. Fyfe, Evolving output codes for multiclass problems, IEEE Transactions of Evolutionary Computation 12 (1) (2007) 93–106.
[49] G. Yule, On the association of attributes in statistics, Philosophical Transactions of the Royal Society of London 194 (1900) 257–319.
[50] L. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine Learning 51 (2) (2003) 181–207.
[51] F. Masulli, G. Valentini, Quantitative evaluation of dependence among outputs in ECOC classifiers using mutual information based measures, in: K. Marko, P. Werbos (Eds.), Proceedings of the International Joint Conference on Neural Networks IJCNN'01, vol. 2, IEEE, Piscataway, NJ, USA, 2001, pp. 784–789.
[52] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, San Mateo, 1993.
[53] K. Crammer, Y. Singer, On the algorithmic implementation of multi-class SVMs, Journal of Machine Learning Research 2 (2001) 265–292.
[54] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Machine Learning 40 (2000) 139–157.