# Subclass Problem-Dependent Design for Error-Correcting Output Codes

Sergio Escalera, David M.J. Tax, Oriol Pujol, Petia Radeva, *Member*, *IEEE*, and Robert P.W. Duin

**Abstract**—A common way to model multiclass classification problems is by means of Error-Correcting Output Codes (ECOCs). Given a multiclass problem, the ECOC technique designs a code word for each class, where each position of the code identifies the membership of the class for a given binary problem. A classification decision is obtained by assigning the label of the class with the closest code. One of the main requirements of the ECOC design is that the base classifier is capable of splitting each subgroup of classes from each binary problem. However, we cannot guarantee that a linear classifier model convex regions. Furthermore, nonlinear classifiers also fail to manage some type of surfaces. In this paper, we present a novel strategy to model multiclass classification problems using subclass information in the ECOC framework. Complex problems are solved by splitting the original set of classes into subclasses and embedding the binary problems in a problem-dependent ECOC design. Experimental results show that the proposed splitting procedure yields a better performance when the class overlap or the distribution of the training objects conceal the decision boundaries for the base classifier. The results are even more significant when one has a sufficiently large training size.

**Index Terms**—Multiclass classification, subclasses, Error-Correcting Output Codes, embedding of dichotomizers.

## 1 Introduction

In the literature, one can find several powerful binary classifiers. However, when one needs to deal with multiclass classification problems, many learning techniques fail to manage this information. Instead, it is common to construct the classifiers to distinguish between just two classes and to combine them in some way. In this sense, Error-Correcting Output Codes (ECOCs) were born as a general framework to combine binary problems to address the multiclass problem. The strategy was introduced by Dietterich and Bakiri [7] in 1995. Based on the error correcting principles [7] and because of its ability to correct the bias and variance errors of the base classifiers [16], ECOC has been successfully applied to a wide range of applications such as face recognition [25], face verification [15], text recognition [12], and manuscript digit classification [27].

The ECOC technique can be broken down into two distinct stages: encoding and decoding. Given a set of classes, the coding stage designs a code word[1] for each class based on different binary problems. The decoding stage makes a classification decision for a given test sample based on the value of the output code.

1. The code word is a sequence of bits of a code representing each class, where each bit identifies the membership of the class for a given binary classifier.

- *S. Escalera, O. Pujol, and P. Radeva are with the Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Barcelona, Spain. E-mail: sergio@maia.ub.es.*
- *D.M.J. Tax and R.P.W. Duin are with the Information and Communication Theory (ICT) Group, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, PO Box 5031, 2600 GA, Delft, The Netherlands. E-mail: d.m.j.tax@gmail.com, r.duin@ieee.org.*

*Manuscript received 14 Sept. 2007; revised 13 Dec. 2007; accepted 4 Jan. 2008; published online 12 Feb. 2008.*
*Recommended for acceptance by J. Matas.*
*For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-09-0599.*
*Digital Object Identifier no. 10.1109/TPAMI.2008.38.*

At the coding step, given a set of $N$ classes to be learned, $n$ different bipartitions (groups of classes) are formed, and $n$ binary problems (dichotomizers) are trained. As a result, a code word of length $n$ is obtained for each class, where each bit of the code corresponds to the response of a given dichotomizer (coded by $+1$, $-1$, according to their class set membership). Arranging the code words as rows of a matrix, we define a *coding matrix* $M$, where $M \in \{-1, 1\}^{N \times n}$ in the binary case. The most well-known binary coding strategies are the one-versus-all strategy [17], where each class is discriminated against the rest of classes, and the dense random strategy [1], where a random matrix $M$ is generated, maximizing the rows and columns separability in terms of the Hamming distance [7]. In Fig. 1a, the one-versus-all ECOC design for a four-class problem is shown. The white regions of the coding matrix $M$ correspond to the positions coded by 1 and the black regions to $-1$. Thus, the code word for class $C_1$ is $\{1, -1, -1, -1\}$. Each column $i$ of the coding matrix codifies a binary problem learned by its corresponding dichotomizer $h_i$. For instance, dichotomizer $h_1$ learns $C_1$ against classes $C_2$, $C_3$, and $C_4$, dichotomizer $h_2$ learns $C_2$ against classes $C_1$, $C_3$, and $C_4$, etc. An example of a dense random matrix for a four-class problem is shown in Fig. 1c.

It was when Allwein et al. [1] introduced a third symbol (the zero symbol) in the coding process when the coding step received special attention. This symbol increases the number of partitions of classes to be considered in a ternary ECOC framework by allowing some classes to be ignored. Then, the ternary coding matrix becomes $M \in \{-1, 0, 1\}^{N \times n}$. In this case, the symbol zero means that a particular class is not considered by a certain binary classifier. Thanks to this, strategies such as one-versus-one [13] and random sparse coding [1] can be formulated in the framework. Fig. 1b shows the one-versus-one ECOC configuration for a four-class problem. In this case, the gray positions correspond to the zero symbol. A possible sparse random matrix for a four-class problem is shown in Fig. 1d. Note that the previous coding designs are predefined. Thus, the training data is not considered until the coding matrix $M$ is constructed. Then,

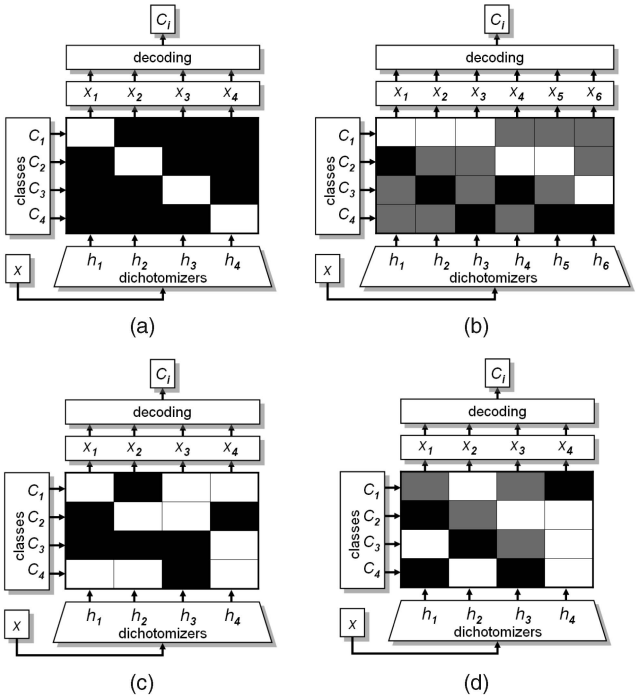



Fig. 1. (a) One-versus-all, (b) one-versus-one, (c) dense random, and (d) sparse random ECOC designs.

each dichotomizer uses the coded positions of $M$ to train the different binary problems.

The decoding step was originally based on error-correcting principles under the assumption that the learning task can be modeled as a communication problem, in which class information is transmitted over a channel [7]. During the decoding process, applying the $n$ binary classifiers, a code is obtained for each data point in the test set. This code is compared to the base code words of each class defined in the matrix $M$, and the data point is assigned to the class with the *closest* code word. The most frequently applied decoding strategies are the Hamming $(HD)$ [17] and the euclidean $(ED)$ decoding distances [13]. With the introduction of the zero symbol, Allwein et al. [1] showed the advantage of using a loss-based function of the margin of the output of the base classifier. Recently, Asuncion and Newman [8] proposed a Loss-Weighted strategy to decode, where a set of probabilities based on the performances of the base classifiers are used to weight the final classification decision. In Fig. 1, each ECOC codification is used to classify an input object $X$. The input data sample $X$ is tested with each dichotomizer $h_i$, obtaining an output $X_i$. The final code $\{X_1, \ldots, X_n\}$ of the test input $X$ is used by a given decoding strategy to obtain the final classification decision. Note that in both, the binary and the ternary ECOC framework, the value of each position $X_j$ of the test code word cannot take the value zero since the output of each dichotomizer $h_j \in \{-1, +1\}$.

Recently, new improvements in the ternary ECOC coding demonstrate the suitability of the ECOC methodology to deal with multiclass classification problems [21], [20], [9], [24], [4]. These recent designs use the knowledge of the problem-domain to learn relevant binary problems from ternary codes. The basic idea of these methods is to use the training data to guide the training process and, thus, to construct the coding

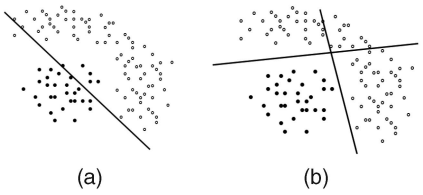



Fig. 2. (a) Decision boundary of a linear classifier of a two-class problem. (b) Decision boundaries of a linear classifier splitting the problem of (a) into two more simple tasks.

matrix $M$, focusing on the binary problems that better fits the decision boundaries of a given data set.

One of the main reasons why the recent problem-dependent designs [21], [20], [9] attains a good performance is because of the high number of possible subgroups of classes that is possible in the ternary ECOC framework. On the other hand, using the training data in the process of the ECOC design allows us to obtain compact code words with high classification performance. However, the final accuracy is still based on the ability of the base classifier to learn each individual problem. Difficult problems, those which the base classifier is not able to find a solution for, require the use of complex classifiers such as Support Vector Machines with Radial Basis Function kernel [18] and expensive parameter optimizations. Look at the example in Fig. 2a. A linear classifier is used to split two classes. In this case, the base classifier is not able to find a convex solution. On the other hand, in Fig. 2b, one of the previous classes has been split into two subsets, which we call *subclasses*. Then, the original problem is solved using two linear classifiers, and the two new subclasses have the same original class label. Some studies in the literature tried to form subclasses using the labels information, which is called Supervised Clustering [26], [5]. In these types of systems, clusters are usually formed without taking into account the behavior of the base classifier that learns the data. In a recent work [28], the authors use the class labels to form the subclasses that improve the performance of particular Discriminant Analysis algorithms.

In this paper, we present a problem-dependent ECOC design, where classes are partitioned into subclasses using a clustering approach for the cases that the base classifier is not capable to distinguish the classes. Sequential Forward Floating Search based on maximizing the Mutual Information is used to generate the subgroups of problems that are split into more simple ones until the base classifier is able to learn the original problem. In this way, multiclass problems that cannot be modeled by using the original set of classes are modeled without the need of using more complex classifiers. The final ECOC design is obtained by combining the subproblems. The novel subclass ECOC design is compared with the state-of-the-art ECOC designs over a set of UCI Machine Learning Repository data sets and on a real multiclass traffic sign categorization problem using different base classifiers. The results show that in most cases the subclass strategy is able to obtain significant performance improvements.

The paper is organized as follows: Section 2 presents the subclass ECOC approach. Section 3 shows the experimental results discussing several points about the subclass ECOC performance. Finally, Section 4 concludes the paper.

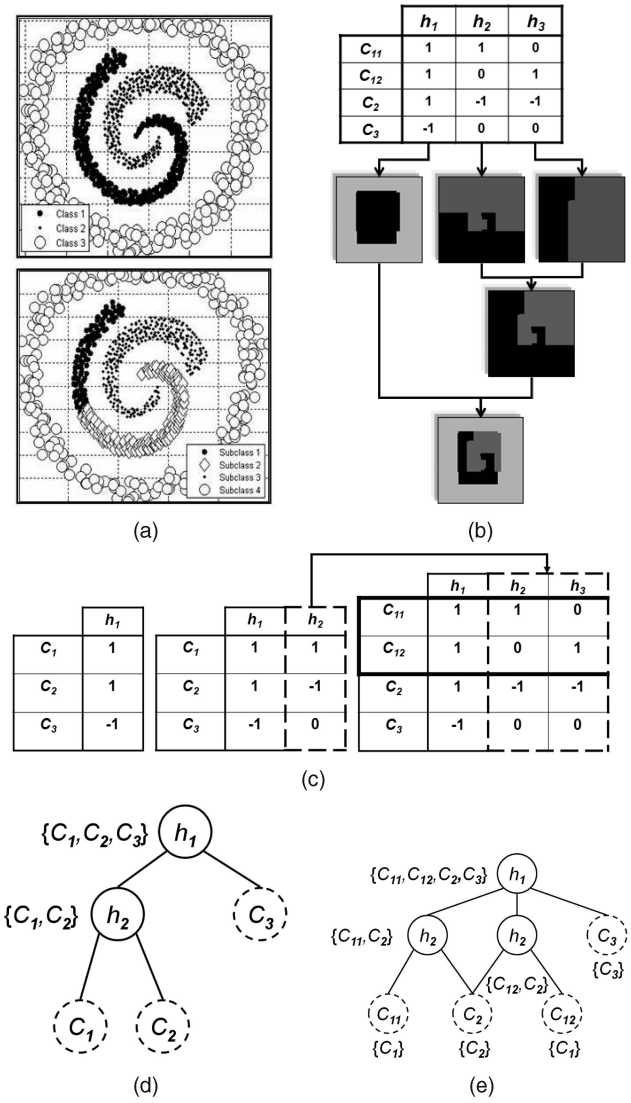



Fig. 3. (a) Top: original three-class problem. Bottom: four subclasses found. (b) Subclass ECOC encoding using the four subclasses using Discrete Adaboost with 40 runs of Decision Stumps. (c) Learning evolution of the subclass matrix $M$. (d) Original tree structure without applying subclass. (e) New tree-based configuration using subclasses.

## 2 Problem-Dependent ECOC Subclass

From an initial set of classes $C$ of a given multiclass problem, the objective of the subclass ECOC strategy is to define a new set of classes $C'$, where $|C'| > |C|$, so that the new set of binary problems is easier to learn for a given base classifier. For this purpose, we use a guided procedure that, in a problem-dependent way, groups classes and splits them into subsets if necessary.

Recently, Ghani [21] proposed a ternary problem-dependent design of ECOC, called DECOC, where given $N$ classes, a high classification performance is achieved with only $N-1$ binary problems. The method is based on the embedding of discriminant tree structures derived from the problem domain. The binary trees are built by looking for the partition that maximizes the *MI* between the data and their respective class labels. Look at the three-class problem shown on the top in Fig. 3a. The standard DECOC algorithm considers the whole set of classes to split it into two subsets of classes $\wp^+$ and $\wp^-$, maximizing the *MI* criterion on a

TABLE 1
Table of Terms

$\theta = \{\theta_{size}, \theta_{perf}, \theta_{impr}\}$ - Parameters for the size, performance, and improvement.
$\zeta_j$ - Centroid of cluster $j$
$C$ - Set of classes
$C'$ - Set of sub-classes
$C_{ij}$ - $j^{th}$ sub-class of class $C_i$
$CD$ - Critical value for statistical significance
$\xi$ - Error function
$h_i$ - $i^{th}$ dichotomizer
$i, j$ - Index
$I$ - Mutual information
$J$ - Data matrix of the original problem
$J_i$ - Data of class $C_i$
$J'$ - Data matrix of the sub-classes
$k$ - Number of methods in the comparative
$L$ - Sets of classes labels
$M$ - Coding matrix
$m$ - Number of object instances
$n$ - Number of dichotomizers
$P$ - Total number of experiments
$N$ - Number of classes
$\wp_i = \{\wp_i^+, \wp_i^-\}$ - Set of positive and negative sub-sets of the $i^{th}$ binary problem
$\rho$ - Objective function
$q_\alpha$ - Studentized range statistic divided by $\sqrt{2}$
$r_i^j$ - Rank of each problem $i$ and each ECOC design $j$
$R_j$ - Mean rank of the $j^{th}$ design

sequential forward floating search (*SFFS*) procedure. In the example, the first subsets found correspond to $\wp^+ = \{C_1, C_2\}$ and $\wp^- = \{C_3\}$. Then, a base classifier is used to train its corresponding dichotomizer $h_1$. This classifier is shown in the node $h_1$ of the tree structure shown in Fig. 3d. The procedure is repeated until all classes are split into separate subsets $\wp$. In the example, the second classifier is trained to split the subsets of classes $\wp^+ = C_1$ from $\wp^- = C_2$ because the classes $C_1$ and $C_2$ were still contained in a single subset after the first step. This second classifier is codified by the node $h_2$ in Fig. 3d. When the tree is constructed, the coding matrix $M$ is obtained by codifying each internal node of the tree as a column of the coding matrix (see Fig. 3c).

In our case, sequential forward floating search (*SFFS*) procedure is also applied to look for the subsets $\wp^+$ and $\wp^-$ that maximizes the MI between the data and their respective class labels [21]. The encoding algorithm is shown in Table 2. The terms used in the paper are summarized in the Table 1.

Given an $N$-class problem, the whole set of classes is used to initialize the set $L$, containing the sets of labels for the classes to be learned. At the beginning of each iteration $k$ of the algorithm (**Step 1**), the first element of $L$ is assigned to $S_k$ in the first step of the algorithm. Next, $SFFS$ is used to find the optimal binary partition $BP$ of $S_k$ that maximizes the MI $I$ between the data and their respective class labels (**Step 2**). The *SFFS* algorithm used [19] is shown in Appendix A, and the implementation details of the fast quadratic MI can be found in Appendix B.

To illustrate our procedure, let us return to the example of the top in Fig. 3a. On the first iteration of the subclass ECOC algorithm, $SFFS$ finds the subset $\wp^+ = \{C_1, C_2\}$ against $\wp^- = \{C_3\}$. The encoding of this problem is shown in the first matrix in Fig. 3c. The positions of the column corresponding to the classes of the first partition are coded by $+1$ and the classes corresponding to the second partition to $-1$, respectively. In our procedure, the base classifier is used to test if the performance obtained by the trained dichotomizers is sufficient. Observe the decision boundaries of the picture next to the first column of the matrix in

TABLE 2
Problem-Dependent Subclass ECOC Algorithm

**Inputs**: $J, C, \theta = \{\theta_{size}, \theta_{perf}, \theta_{impr}\}$ //Thresholds for the number of samples, performance, and improvement between iterations
**Outputs**: $C', J', \wp', M$
**[Initialization:]**
Create the trivial partition $\{\wp_0^+, \wp_0^-\}$ of the set of classes $\{C_i\}$:
$\{\wp_0^+, \wp_0^-\} = \{\{\emptyset\}, \{C_1, C_2, ..., C_N\}\}$
$L_0 = \{\wp_0^-\}; J' = J; C' = C; \wp' = \emptyset; M = \emptyset; k = 1$
**Step 1** $S_k$ is the first element of $L_{k-1}$
$L'_k = L_{k-1} \backslash \{S_k\}$
**Step 2** Find the optimal binary partition $BP(S_k)$:
$\{\wp_k^+, \wp_k^-\} = argmax_{BP(S_k)}(I(\mathbf{x}, d(BP(S_k))))$
where $I$ is the mutual information criterion, $\mathbf{x}$ is the random variable associated to the features and $d$ is the discrete random variable of the dichotomy labels[a], defined in the following terms,

$$d = d(\mathbf{x}, BP(S_k)) = \begin{array}{ll} 1 & \text{if } \mathbf{x} \in C_i | C_i \in \wp_k^+ \\ -1 & \text{if } \mathbf{x} \in C_i | C_i \in \wp_k^- \end{array}$$

**Step 3** // Look for sub-classes
$\{C', J', \wp'\} = SPLIT(J_{p_k^+}, J_{p_k^-}, C', J', J, \wp', \theta)^b$
**Step 4** $L_k = \{L'_k \cup \wp_k^i\}$ if $|\wp_k^i| > 1 \ \forall i \in \{+, -\}$
**Step 5** If $|L_k| \neq 0$
$k = k + 1$ **go to Step 1**
**Step 6** Codify the coding matrix $M$ using each partition $\{\wp_i^+, \wp_i^-\}$ of $\wp', i \in [1, .., |\wp'|]$ and each class $C_r \in \wp_i = \{\wp_i^+ \cup \wp_i^-\}$ as follows:

$$M(C_r, i) = \begin{cases} 0 & \text{if } C_r \notin \wp_i \\ +1 & \text{if } C_r \in \wp_i^+ \\ -1 & \text{if } C_r \in \wp_i^- \end{cases} \quad (1)$$

[a]Use $SFFS$ of Appendix A as the maximization procedure and $MI$ of Appendix B to estimate $I$
[b]Using the splitting algorithm of Table 3.

TABLE 3
Subclass *SPLIT* Algorithm

**Inputs**: $J_{\wp^1}, J_{\wp^2}, C', J', J, \wp', \theta$ // $C'$ is the final set of classes, $J'$ the data for the final set of classes, and $\wp'$ is the labels for all the partitions of classes of the final set.
**Outputs**: $C', J', \wp'$
**Step 1** Split problems:
$\{J_{\wp^+}^+, J_{\wp^+}^-\} = SC(J_{\wp^+})^a$
$\{J_{\wp^-}^+, J_{\wp^-}^-\} = SC(J_{\wp^-})$
**Step 2** Select sub-classes:
if $|J_{\wp^+}^+, J_{\wp^+}^-| > |J_{\wp^-}^+, J_{\wp^-}^-|$ // find the largest distance between the means of each sub-set.
$\{J_+^+, J_+^-\} = \{J_{\wp^+}^+, J_{\wp^-}\}; \{J_-^+, J_-^-\} = \{J_{\wp^+}^-, J_{\wp^-}\}$
else
$\{J_+^+, J_+^-\} = \{J_{\wp^-}^+, J_{\wp^+}\}; \{J_-^+, J_-^-\} = \{J_{\wp^-}^-, J_{\wp^+}\}$
end
**Step 3** Test parameters to continue splitting:
if $TEST\_PARAMETERS(J_{\wp^1}, J_{\wp^2}, J_1^1, J_1^2, J_2^1, J_2^2, \theta)$// call the function with the new sub-sets
$\{C', J', \wp'\} = SPLIT(J_1^1, J_1^2, C', J', J, \wp', \theta)$
$\{C', J', \wp'\} = SPLIT(J_2^1, J_2^2, C', J', J, \wp', \theta)$
end
**Step 4** Save the current partition:
Update the data for the new sub-classes and previous sub-classes if intersections exists $J'$.
Update the final number of sub-classes $C'$.
Create $\wp_c = \{\wp_{c^1}, \wp_{c^2}\}$ the set of labels of the current partition.
Update the labels of the previous partitions $\wp$.
Update the set of partitions labels with the new partition $\wp' = \wp' \cup \wp_c$.

[a]$SC$ corresponds to the splitting method of the input data into two main clusters.

Fig. 3b. One can see that the base classifier finds a good solution for this first problem.

Then, the second classifier is trained to split $\wp^+ = C_1$ against $\wp^- = C_2$, and its performance is computed. To separate the current subsets is not a trivial problem, and the classification performance is poor. Therefore, our procedure tries to split the data $J_{\wp^+}$ and $J_{\wp^-}$ from the current subsets $\wp^+$ and $\wp^-$ into more simple subsets. At **Step 3** of the algorithm, the splitting criteria $SC$ takes as input a data set $J_{\wp^+}$ or $J_{\wp^-}$ from a subset $\wp^+$ or $\wp^-$ and splits it into two subsets $J_{\wp^+}^+$ and $J_{\wp^+}^-$ or $J_{\wp^-}^+$ and $J_{\wp^-}^-$. In Section 3, we discuss the selection of the splitting criterion. The splitting algorithm is shown in Table 3.

When two data subsets $\{J_{\wp^+}^+, J_{\wp^+}^-\}$ and $\{J_{\wp^-}^+, J_{\wp^-}^-\}$ are obtained, only one of both split subsets is used. We select the subsets that have the highest distance between the means of each cluster. Suppose that the distance between $J_{\wp^-}^+$ and $J_{\wp^-}^-$ is larger than between $J_{\wp^+}^+$ and $J_{\wp^+}^-$. Then, only $J_{\wp^+}$, $J_{\wp^-}^+$, and $J_{\wp^-}^-$ are used. If the new subsets improve the classification performance, new subclasses are formed, and the process is repeated.

In the example in Fig. 3, applying the splitting criteria $SC$ over the two subsets, two clusters are found for $\wp^+ = C_1$ and for $\wp^- = C_2$. Then, the original encoding of the problem $C_1$ versus $C_2$ (corresponding to the second column of the matrix in the center in Fig. 3c) is split into two columns marked with the dashed lines in the matrix on the right. In this way, the original $C_1$ versus $C_2$ problem is transformed to two more simple problems $\{C_{11}\}$ against $\{C_2\}$ and $\{C_{12}\}$ against $\{C_2\}$. Here, the first subindex of the class corresponds to the original class, and the second subindex to the number of subclass. It implies that the class $C_1$ is split into two subclasses (look at the bottom of Fig. 3a), and the original three-class problem $C = \{C_1, C_2, C_3\}$ becomes the four-subclass problem $C' = \{C_{11}, C_{12}, C_2, C_3\}$. As the class $C_1$ has been decomposed by the splitting of the second problem, we need to save the information of the current subsets and the previous subsets affected by the new splitting. The steps to update this information are summarized in the **Step 4** of the splitting algorithm. We use the object labels to define the set of subclasses of the current partition $\wp_c$. If new subclasses are created, the set of subclasses $C'$ and the data for subclasses $J'$ have to be updated. Note that when a class or a subclass previously considered for a given binary problem is split in a future iteration of the procedure, the labels from the previous subsets $\{\wp^+, \wp^-\}$ need to be updated with the new information. Finally, the set of labels for the binary problems $\wp'$ is updated with the labels of the current subset $\wp' = \wp' \cup \wp_c$. In the example in Fig. 3, the dichotomizer $h_1$ considers the subsets $\wp_1^+ = \{C_1, C_2\}$ and $\wp_1^- = \{C_3\}$. Then, those positions containing class $C_1$ are replaced with $C_{11}$ and $C_{12}$. The process is repeated until the desired performance is achieved, or the stopping conditions are fulfilled.

The conditions that guide the learning and splitting process are defined by the set of parameters $\theta = \{\theta_{size}, \theta_{perf}, \theta_{impr}\}$, where $\theta_{size}$ corresponds to the minimum size of a subset to be clustered, $\theta_{perf}$ contains the minimum error desired for each binary problem, and $\theta_{impr}$ looks for the improvement of the split subsets regarding the previous ones. The function $TEST\_PARAMETERS$ in Table 3 is responsible for testing the constraints based on the parameters $\{\theta_{size}, \theta_{perf}, \theta_{impr}\}$. If the constraints are satisfied, the new subsets are selected and used to recursively call the splitting function (**Step 3** of the algorithm in Table 3). The constraints of the function $TEST\_PARAMETERS$ are fixed by default as follows:

- The number of objects in $J_{\wp^+}$ has to be larger than $\theta_{size}$.
- The number of objects in $J_{\wp^-}$ has to be larger than $\theta_{size}$.

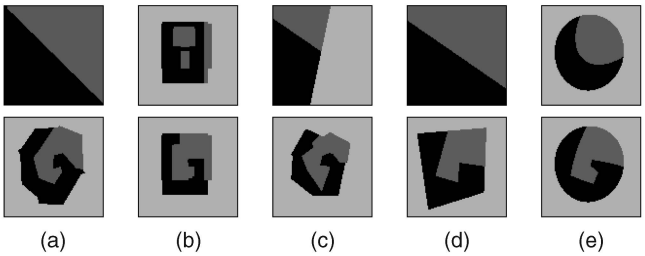



Fig. 4. Subclass ECOC without subclasses (top) and including subclasses (bottom). (a) *FLDA*. (b) Discrete Adaboost. (c) *NMC*. (d) Linear *SVM*. (e) *RBF SVM*.

- The error $\xi(h(J_{\wp^-}, J_{\wp^+}))$ obtained from the dichotomizer $h$ using a particular base classifier applied on the sets $\{\wp^+, \wp^-\}$ has to be larger than $\theta_{perf}$.
- The sum of the well-classified objects from the two new problems (based on the confusion matrices) divided by the total number of objects has to be greater than $1 - \theta_{impr}$.

$\theta_{size}$ avoids the learning of very unbalanced problems. $\theta_{perf}$ determines when the performance of a partition of classes is insufficient, and subclasses are required. Moreover, finally, when a partition does not obtain the desired performance $\theta_{perf}$, the splitting of the data stops, preventing overtraining.

In the example in Fig. 3, the three dichotomizers $h_1$, $h_2$, and $h_3$ find a solution for the problem (look the trained boundaries shown in Fig. 3b), obtaining a classification error under $\theta_{perf}$, so, the process stops. Now, the original tree encoding of the DECOC design shown in Fig. 3d can be represented by the tree structure in Fig. 3e, where the original class associated to each subclass is shown in the leaves.

Summarizing, when a set of objects belonging to different classes is split, object labels are not taken into account. It can be seen as a clustering in the sense that the subsets are split into more simple ones while the splitting constraints are satisfied. It is important to note that when one uses different base classifiers, the subclass splitting is probably applied to different classes or subclasses and, therefore, the final number of subclasses and binary problems differs.

When the final set of binary problems is obtained, its respective set of labels $\wp'$ is used to create the coding matrix $M$ (1). The outputs $C'$ and $J'$ contain the final set of subclasses, and the new data for each subclass, respectively.

Finally, to decode the new subclass problem-dependent design of ECOC, we take advantage of the recently proposed Loss-Weighted decoding design [8]. The decoding strategy uses a set of normalized probabilities based on the performance of the base classifier and the ternary ECOC constraints [8]. The decoding algorithm is described in Appendix C.

### 2.1 Illustration over Toy Problems

To show the effect of the subclass ECOC strategy for different base classifiers, we used the previous toy problem of the top in Fig. 3a. Five different base classifiers are applied: Fisher Linear Discriminant Analysis (*FLDA*), Discrete Adaboost, Nearest Mean Classifier, Linear *SVM*, and *SVM* with Radial Basis Function kernel.[2] Using these base classifiers on the toy problem, the original DECOC strategy with the Loss-Weighted algorithm obtains the decision boundaries shown on the top row in Fig. 4. The new learned boundaries are shown on the bottom row in Fig. 4 for fixed parameters $\theta$. Depending on the flexibility of the base classifier more subclasses are required and, thus, more binary problems. Observe that all base classifiers are able to find a solution for the problem, although with different types of decision boundaries.

2. The parameters of the base classifiers are explained in the experimental results section.

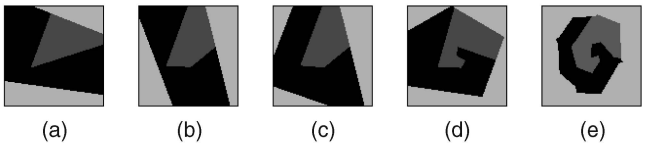



Fig. 5. Learned boundaries using $FLDA$ with $\theta_{size} = \frac{|J|}{50}$, $\theta_{impr} = 0.95$, and (a) $\theta_{perf} = 0.2$, (b) $\theta_{perf} = 0.15$, (c) $\theta_{perf} = 0.1$, (d) $\theta_{perf} = 0.05$, and (e) $\theta_{perf} = 0$, respectively.

TABLE 4
UCI Machine Learning Repository Data Sets Characteristics

| Problem | #Train | #Attributes | #Classes |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Ecoli | 336 | 8 | 8 |
| Wine | 178 | 13 | 3 |
| Glass | 214 | 9 | 7 |
| Thyroid | 215 | 5 | 3 |
| Vowel | 990 | 10 | 11 |
| Balance | 625 | 4 | 3 |
| Yeast | 1484 | 8 | 10 |

The selection of the set of parameters $\theta$ has a decisive influence on the final results. We can decrease the value of $\theta_{perf}$ and increase the value of $\theta_{impr}$ to obtain a better solution for a problem, but we need to optimize the parameters to avoid overtraining by stopping the procedure if no more improvement can be achieved. In the same way, sometimes to obtain the best solution for a problem implies to learn more simple problems. These points should be considered to obtain the desired trade-off between performance and computational cost. A simple example to show the evolution of learning for different parameters $\theta$ over the previous problem is shown in Fig. 5. The base classifier applied is $FLDA$. One can observe that when $\theta_{perf}$ decreases, more dichotomizers are required to obtain a higher performance. Thus, to achieve the desired accuracy, more subclasses and binary problems are needed.

## 3 Experimental Results

In this section, we compare the subclass approach with different state-of-the-art coding designs and base classifiers on real and synthetic data sets. In order to evaluate the methodology, first, we discuss the data, compared methods, experiments, and performance evaluation.

- *Data*. The data used for the experiments consists of eight arbitrary multiclass data sets from the UCI Machine Learning Repository [2] and one real nine-class traffic sign classification problem from the Geomobil project of [3]. The characteristics of the UCI data sets are shown in Table 4. It shows a wide range of attributes sizes and class sizes. All data sets have been normalized with respect to the mean and variance.

- *Compared methods*. We compare our method with the state-of-the-art ECOC coding designs: one-versus-one [13], one-versus-all [17], dense random [1], sparse random [1], and DECOC [21].

  The random matrices were selected from a set of 20,000 randomly generated matrices, with $P(1) = P(-1) = 0.5$ for the dense random matrix and $P(1) = P(-1) = P(0) = 1/3$ for the sparse random matrix. The number of binary problems was fixed to the number of classes. Therefore, a direct comparison to the one-versus-all and DECOC designs is possible. Each strategy uses the previously mentioned Linear Loss-weighted decoding to evaluate their performances at identical conditions. Five different base classifiers are applied over each ECOC configuration: Nearest Mean Classifier (*NMC*) with the classification decision using the euclidean distance between the mean of the classes, Discrete Adaboost with 40 iterations of Decision Stumps [11], Linear Discriminant Analysis implementation of the PR Tools using the default values [10], OSU implementation of Linear Support Vector Machines with the regularization parameter $C$ set to 1 [18], and OSU implementation of Support Vector Machines with Radial Basis Function kernel with the default values of the regularization parameter $C$ and the gamma parameter set to 1 [18].[3]
- *Experiments*. First, we classify the set of UCI Machine Learning Repository data sets with the ECOC designs and the different base classifiers. Second, focusing on particular data sets, we analyze the performance of our methodology over the training and test sets by changing the values of the set of parameters $\theta$. Furthermore, we also perform an experiment to show the behavior of our procedure when working with different training sizes. Third, a real multiclass traffic sign recognition problem is evaluated using the different strategies.
- *Performance evaluation*. To evaluate the performance of the different experiments, we apply stratified 10-fold cross validation and test for the confidence interval at 95 percent with a two-tailed t-test. Besides, we use the statistical Nemenyi test to look for significant differences between the method performances [6].

## 3.1 UCI Machine Learning Repository

Using the UCI Machine Learning Repository data sets, we perform different experiments. First, we classify the eight data sets. Second, we look for the statistical significance of the results and, finally, we discuss the effect of the subclass parameters and the training size.

### 3.1.1 UCI Machine Learning Repository Classification

Using the UCI data sets in Table 4, the five base classifiers, and the six ECOC designs, we have performed a total of 240 10-fold tests. The set of parameters of the subclass approach $\theta = \{\theta_{size}, \theta_{perf}, \theta_{impr}\}$ has been fixed to $\theta_{size} = \frac{|J|}{50}$ minimum number of objects to apply subclass (thus, 2 percent of the samples of each particular problem), $\theta_{perf} = 0$ to split classes if the binary problem does not learn properly the training objects, and $\theta_{impr} = 0.95$, which means that the split problems must improve at least a 5 percent of the performance of the problem without splitting. The last measure is simply estimated by dividing the sum of the well-classified objects from the two subproblems the total number of objects by looking at the confusion matrices. For simplicity and fast computation, the used splitting criterion is K-Means with $k = 2$. K-Means is a fast way to split a set of objects into $k$ clusters satisfying intracluster compactness and high intercluster separability by minimizing an objective function

$$\rho = \sum_{i=1}^{2} \sum_{j=1}^{m} \left\| x_i^j - \zeta_j \right\|^2, \tag{2}$$

for $m$ object instances, $\zeta_j$ is the centroid for the cluster $i \in \{1, 2\}$, and $\rho$ is the objective function to be minimized.[4]

The results for each base classifier are shown in the tables of Appendix D. In Figs. 6 and 7, the results are graphically illustrated for the Discrete Adaboost and *NMC* classifiers, respectively.

These two base classifiers obtain the least and most performance improvements, respectively. Although the results for Adaboost show that the subclass approach is comparable with the other ECOC approaches, it cannot be considered statistically significantly better. It is caused by the fact that Adaboost is a relatively strong classifier, and it is able to fit better the problem boundaries. On the other hand, looking at the results in Fig. 7, one can see that the results of the subclass approach are significantly better for most of the cases because of the failure of *NMC* to model the problems by only using the original set of classes.

### 3.1.2 Statistical Significance

To check for statistically significant differences between the methods, we use the Nemenyi test—two techniques are significantly different when the corresponding average ranks differ by at least the critical difference value. The ranks are obtained estimating each particular rank $r_i^j$ for each problem $i$ and each ECOC design $j$, and then, computing the mean rank $R$ for each design as $R_j = \frac{1}{P}\sum_i r_i^j$, being $P$ the number of experiments. The mean rank of each ECOC design for each base classifier and for the whole set of problems are numerically shown in Table 5.[5] The critical value $(CD)$ [6] is defined as follows:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6P}}, \tag{3}$$

where $q_\alpha$ is based on the Studentized range statistic divided by $\sqrt{2}$, $k = 6$ is the number of methods in the comparison, and $P = 40$ is the total number of experiments performed (8 data sets $\times 5$ base classifiers). In our case, when comparing six methods with a confidence value $\alpha = 0.05$, $q_{0.05} = 2.20$. Substituting this in (3), we obtain a critical difference value of 0.81.

3. The regularization parameter $C$ and the gamma parameter are set to 1 for all the experiments. We selected this parameter after a preliminary set of experiments. We decided to keep the parameter fixed for the sake of simplicity and easiness of replication of the experiments, though we are aware that this parameter might not be optimal for all data sets. Nevertheless, since the parameters are the same for all the compared methods, any weakness in the results will also be shared.

4. It is important to save the history of splits to reuse the subgroups if they are required again. It speeds up the method and also reduces the variation in the results induced by different random initializations of K-Means.

5. We realize that averaging over data sets has a very limited meaning as it entirely depends on the selected set of problems.

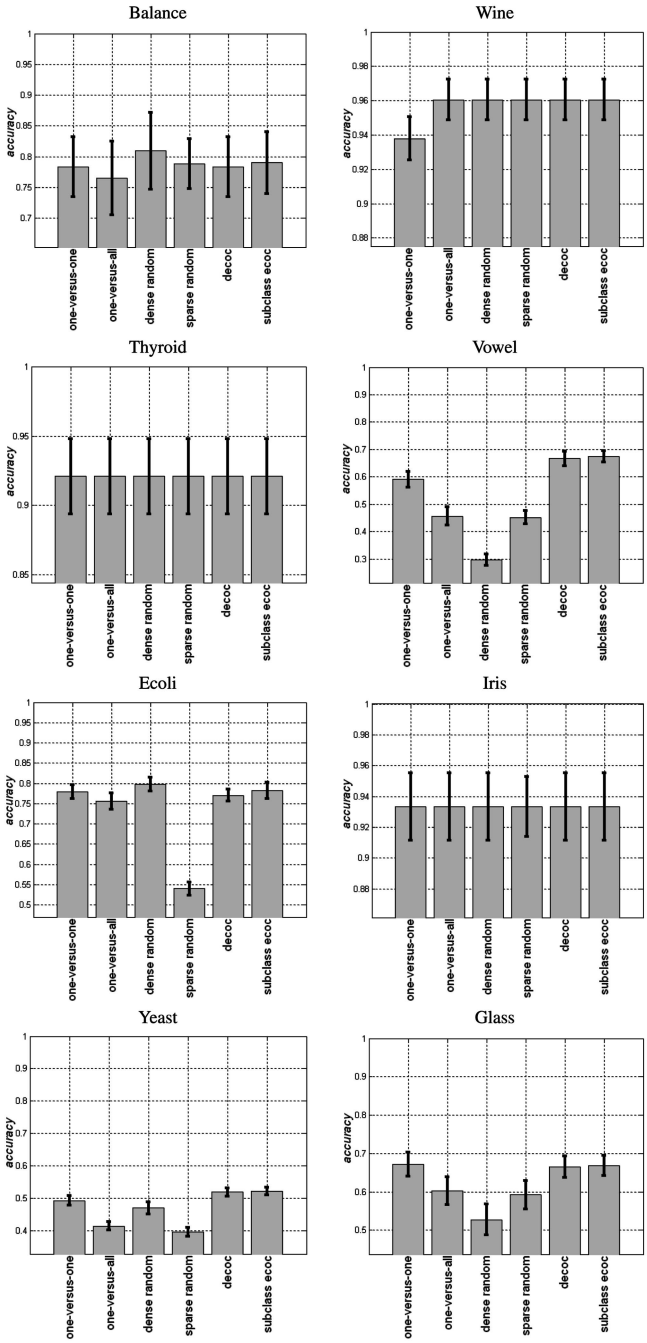



Fig. 6. UCI experiments for Discrete Adaboost.

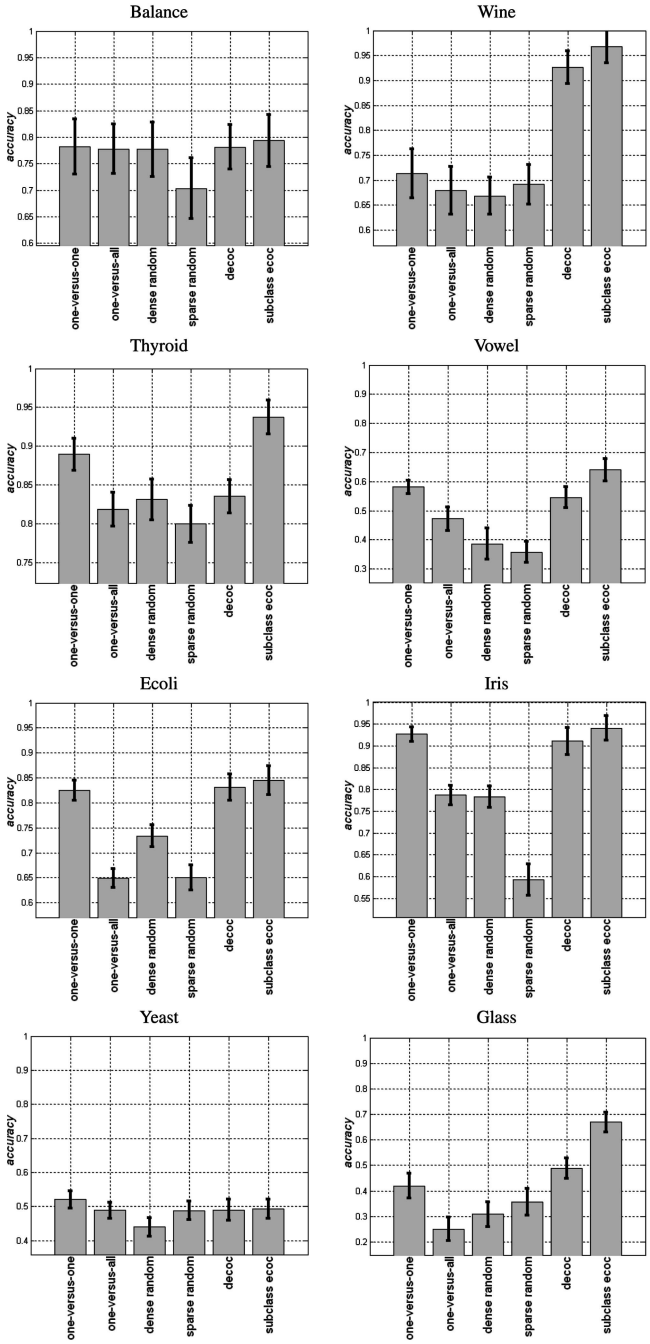



Fig. 7. UCI experiments for *NMC*.

Observing the ranks of each ECOC design in the global rank row in Table 5, one can observe that there are no combinations of methods for which the difference is smaller than the critical value of 0.81, and therefore, we can argue that the subclass approach is significantly better at 95 percent of the confidence interval in the present experiments.

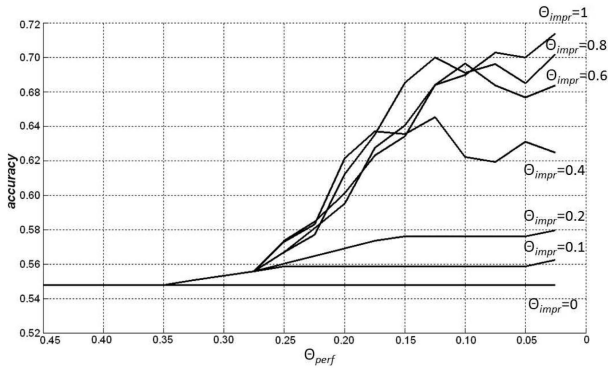
### 3.1.3 Parameters and Training Size

To show the effect of changing the parameters $\theta$, we performed an experiment using the UCI Glass data set. In this experiment, the parameter $\theta_{size}$ is fixed to $\frac{|J|}{50}$, and the values for $\theta_{perf}$ are varied between 0.45 and 0 decreasing by 0.025 per step. For each value of $\theta_{perf}$, the values for $\theta_{impr}$ are {0, 0.1, 0.2, 0.4, 0.6, 0.8, 1}.

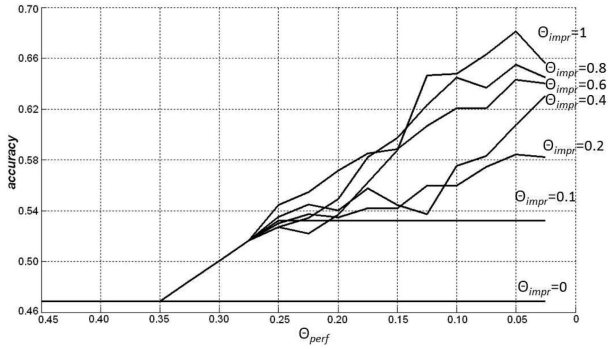
The results of these experiments using $NMC$ as the base classifier are shown graphically in Fig. 8. In this particular problem, one can observe that until the value of $\theta_{perf} = 0.35$, the subclass is not required since all the binary problems achieve a performance greater than this value. When this value is decreased, binary problems require the subclass splitting approach. When $\theta_{impr}$ is increased, both the training and test performance increase. One can also observe that in the case of values near 0 for $\theta_{size}$ and near 1 for $\theta_{perf}$, the system can specialize into very small problems, which results in overtraining. This phenomenon is just visible for $\theta_{perf} = 0.025$ and high values for $\theta_{impr}$ on the test set.

TABLE 5
Rank Positions of the Classification Strategies for the UCI Experiments

| | 1-vs-1 | 1-vs-all | dense | sparse | DECOC | Sub-class |
|---|---|---|---|---|---|---|
| Adaboost | 2.2 | 3.2 | 2.6 | 3.5 | 2.2 | **1.3** |
| NMC | 2.2 | 4.7 | 5.0 | 5.2 | 2.6 | **1.1** |
| FLDA | 1.6 | 3.8 | 3.1 | 3.8 | 2.1 | **1.3** |
| Linear SVM | 2.1 | 3.5 | 3.3 | 3.2 | 1.8 | **1.0** |
| RBF SVM | 2.3 | 4.2 | 2.6 | 4.3 | 2.6 | **1.2** |
| Global rank | 2.1 | 3.9 | 3.3 | 4.0 | 2.3 | **1.2** |



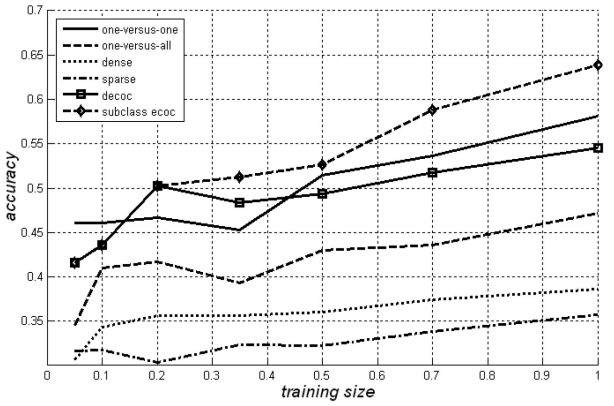

(a)




(b)

Fig. 8. Comparison of the subclass ECOC performances using $NMC$ on the UCI Glass data set for different parameters $\theta_{perf}$ and $\theta_{impr}$. (a) Training set. (b) Test set.

Furthermore, we analyzed the behavior of the present methodology when the training size is changed. In particular, we selected the 11-class Vowel UCI data set and performed 10-fold classification using the different ECOC designs for different training sizes: 5, 10, 20, 35, 50, 70, and 100 percent of the training size. The base classifier in this case is also $NMC$. The results are shown in Fig. 9a. The mean number of subclasses and binary problems is shown in the table in Fig. 9b. One can observe that for small training sizes, the subclass ECOC does not split classes. At these first stages, the subclass ECOC becomes the DECOC strategy, and the performance is also similar, even inferior, to the one obtained by the one-versus-one strategy. When the training size is increased, though the general behavior for all the strategies is to increase their performance, the subclass ECOC is the one, which the improvement is the most significant. Note that the performance improvement of the subclass strategy also increases the number of subclasses and binary problems. Still, the mean number of binary problems of 24.7 is significantly less than the 55 required for the one-versus-one strategy.




(a)

| 5 | 10 | 20 | 35 | 50 | 70 | 100 |
|---|---|---|---|---|---|---|
| 11 × 10 | 11 × 10 | 11.5 × 10.3 | 12.3 × 13.7 | 15.5 × 16.4 | 18.2 × 19.1 | 22.3 × 24.7 |

(b)

Fig. 9. (a) Test performances for the Vowel UCI data set for different percentages of the training size. (b) Mean number of subclasses and binary problems estimated by the subclass ECOC for each training size. The confidence intervals of the results are between 1 percent and 2 percent.

Fig. 10. Speed data set samples.

TABLE 6
Rank Positions of the Classification Strategies for the Speed Data Set

| | 1-vs-1 | 1-vs-all | dense | sparse | DECOC | Sub-class |
|---|---|---|---|---|---|---|
| Adaboost | 66.1(3.1) | 56.6(3.1) | 55.2(2.8) | 52.3(3.6) | 58.6(3.2) | 60.8(3.1) |
| NMC | 60.7(3.2) | 50.65(3.7) | 47.4(3.8) | 45.1(3.8) | 51.9(3.2) | 62.8(3.1) |
| FLDA | 74.7(2.8) | 71.4(2.9) | 74.9(2.6) | 72.7(2.5) | 72.6(2.8) | 76.2(3.0) |
| Linear SVM | 74.9(2.7) | 72.3(2.1) | 71.8(2.1) | 68.2(2.9) | 78.9(2.1) | 78.9(1.9) |
| RBF SVM | 45.0(0.9) | 45.0(0.9) | 45.0(0.9) | 44.0(0.9) | 45.0(0.9) | 45.0(0.9) |
| Global rank | 1.8 | 3.6 | 3.4 | 4.6 | 2.6 | 1.2 |

## 3.2 Traffic Sign Categorization

For this experiment, we use the video sequences obtained from the Mobile Mapping System [3] to test a real traffic sign categorization problem. We choose the speed data set since the low resolution of the image, the noncontrolled conditions, and the high similarity among classes make the categorization a difficult task. In this system, the position and orientation of the different traffic signs are measured with fixed video cameras in a moving vehicle. The system has a stereo pair of calibrated cameras, which are synchronized with a GPS/INS system. The result of the acquisition step is a set of stereo-pairs of images with their position and orientation information. Fig. 10 shows several samples of the speed data set used for the experiments. The data set contains a total of 2,500 samples divided into nine classes. Each sample is composed of 1,200 pixel-based features after smoothing the image and applying a histogram equalization. From this original feature space, about 150 features are derived using a $PCA$ that retained 90 percent of the total variance.

The performance and the estimated ranks using the different ECOC strategies for the different base classifiers are shown in Table 6. These results are also illustrated in the

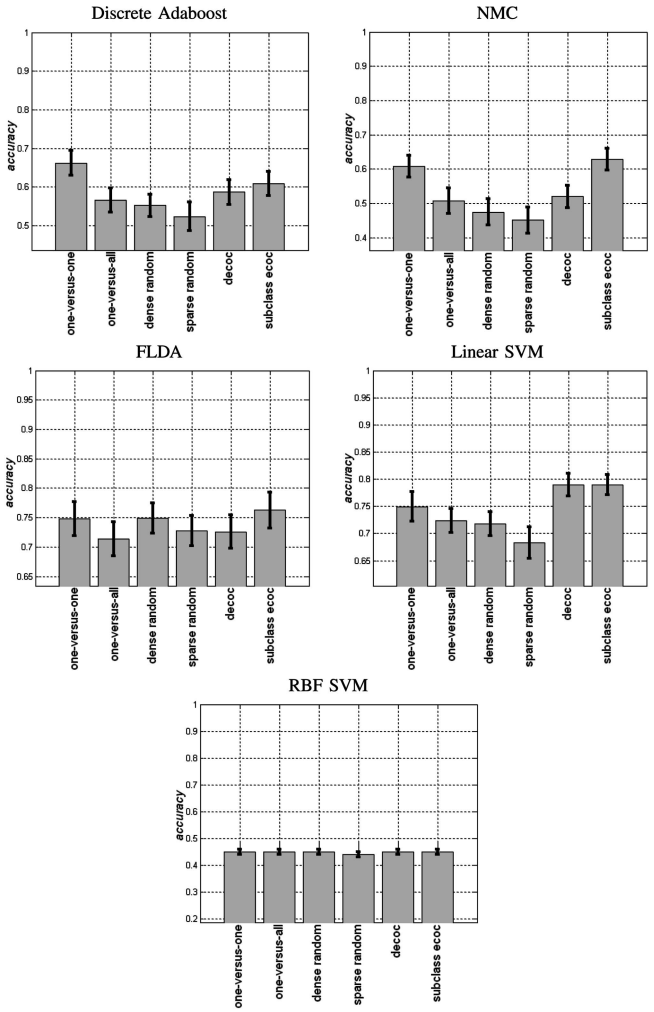



Fig. 11. Speed data set performances.

graphics in Fig. 11. One can see that in this particular problem, the subclass is only required for Discrete Adaboost and $NMC$, while the rest of base classifiers are able to find a solution for the training set without the need for subclasses. In this case, *RBF SVM* obtains low performances, and parameter optimization should be applied to improve these results. Nevertheless, it is out of the scope of this paper. Finally, though the results do not significantly differ between the strategies, the subclass ECOC approach attains a better position in the global rank in Table 6.

### 3.3 Discussions

The subclass ECOC presents an alternative to the use of complex base classifiers. Still, some issues need to be discussed.

As a consequence of applying the subclass strategy, in the worst case, it remains the same than without using subclasses. One of the important points is that both, base classifier and subclass, can be optimized. If the base classifier is well tuned, less binary problems and subclasses would be required by the subclass strategy. On the other hand, the subclass approach could be seen as an incremental tool independent of the base classifier to improve the weakness of the base classifiers. For example, none of the variants of

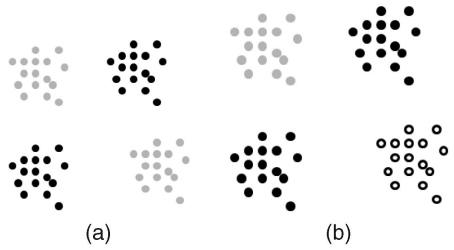



Fig. 12. (a) Original distribution of data for two classes. (b) First subclass splitting.

Adaboost with decision stumps is able to model the XOR problem shown in Fig. 12a. Fig. 12b shows the first splitting of classes found by the subclass strategy. In this case, Adaboost is able to model a solution considering partitions of problems using the three new subclasses. Thus, we can take advantage from both, optimizing a base classifier and optimizing the subclass approach.

Regarding the space of parameters $\theta$, in the present experiments the subclass ECOC obtains significantly better performance with fixed parameters. The fixed parameters have been chosen after a preliminary set of experiments. If it is required, one can also look for the optimum set $\theta$, which attains the best performance by using, for example, cross validation over the training or validation sets.

Similarly, the K-Means splitting criterion (that was used as a simple clustering strategy for the sake of simplicity and fast computation) can be replaced by another criterion. Suppose that we decide to keep the base classifier to be linear. In that case, it is more desirable to adapt the clustering strategy so that it guarantees the linear separability of the new clusters.

In Fig. 13, we performed the classification of UCI data sets for the subclass strategy and the different base classifiers changing the splitting criterion. We compared the K-Means splitting with a hierarchical clustering and the Graph Cut clustering [22]. The two clusters of the hierarchical tree are estimated using euclidean distance to centroid for linkage. The results show that the behavior of three strategies are very similar, and there are no significant differences on the final performances.

Obviously, when we have the new split of classes, an important consideration is to decide if they contribute to improve the accuracy of the whole system. At this point, $\theta_{impr}$ looks for the improvement of the new subclasses respect to the previous group without splitting. We commented that the value of $\theta_{impr}$ has been selected to obtain reasonably good results after a previous set of experiments. A good selection of this parameter is crucial for a good generalization of the strategy. Note that if we fix $\theta_{impr}$ to require a high-performance improvement, in some cases, the system could not gain from subclasses. On the other hand, a small improvement could make the system to focus on very small problems, which really do not contribute to the whole performance and that can produce overtraining.

To look for the minimum $\theta_{impr}$, the implemented heuristic is based on the errors of the confusion matrix for the split groups. It provides a fast computation. Of course, the heuristic may be not optimal, and different strategies can be used instead. A natural way to deal with this problem is to

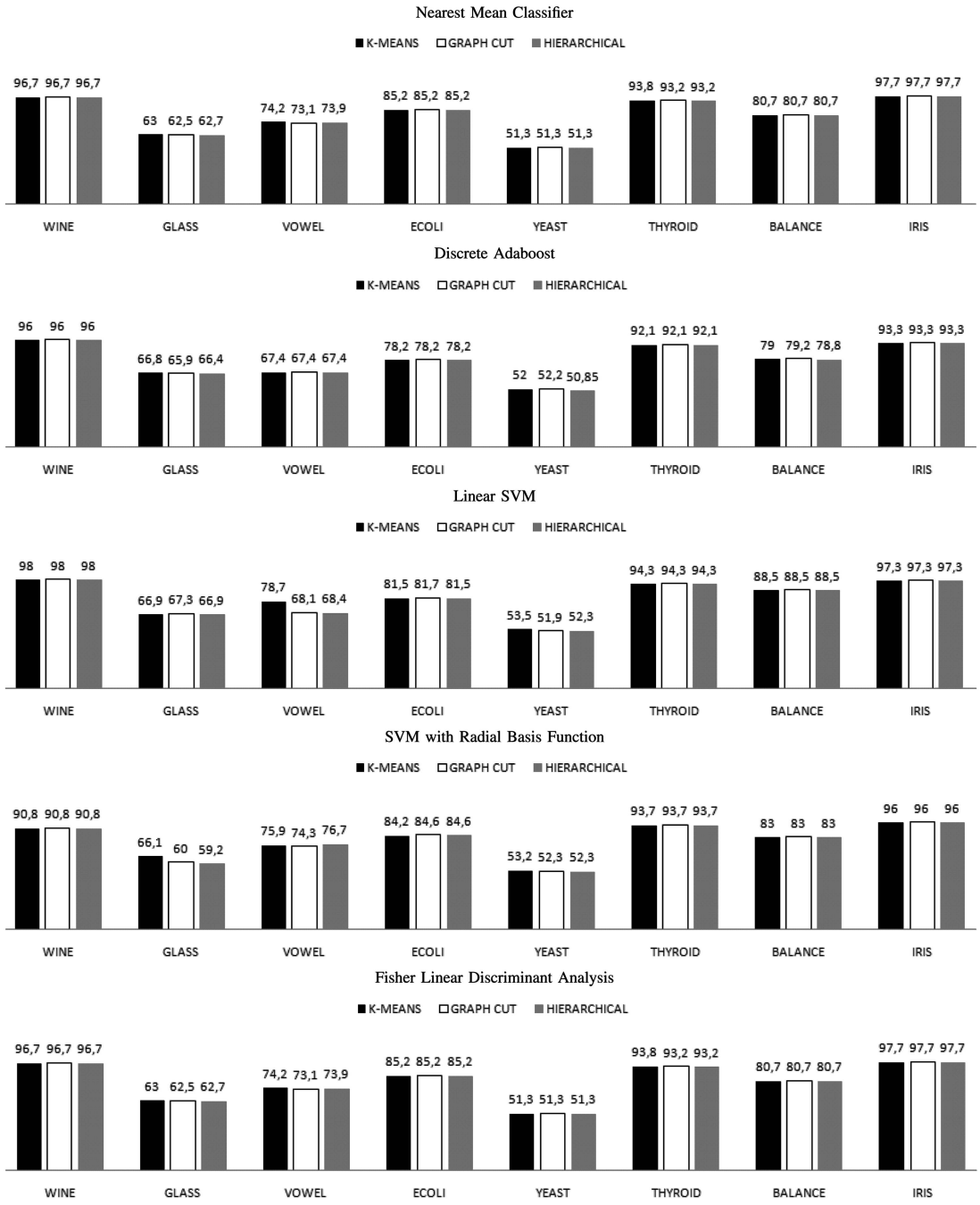



Fig. 13. Classification performance on UCI data sets for subclass ECOC strategy with different splitting criteria.

look for the improvement of the whole system when including the new split problems (thus, evaluating the whole subclass ECOC system each time that a new problem is tested to be included). Testing this heuristic, we found that the obtained performance was very similar to the one obtained by the former approach, still maintaining a similar final number of subclasses and binary problems but considerably increasing the computational cost.

Finally, it is important to make clear that the subclass scheme not only can be used in the present methodology. The final subclasses can be potentially useful to improve other multiclass strategies or standard hierarchical clustering strategy. Note that the final set of $N$ subclasses can be used, for example, over the one-versus-one ECOC configuration. Obviously, it will require a high number of dichotomizers to codify the problem, but in cases where the computational cost is not a problem, it could result a promising choice.

All the comparisons of this paper are performed inside the ECOC framework. We did not compare with other combining strategies or advanced classifiers such as Neural Networks or multiclass Support Vector Machines. This will require a much more extensively study.

## 4 Conclusions

The subclass ECOC strategy presents a novel way to model complex multiclass classification problems. The method is

TABLE 7
SFFS Algorithm

**Input**:
$Y = \{x_j | j = 1..D\}$//Avaliable items//
**Output**:
$X_k = \{x_j | j = 1..|Y|(or D), x_j \in Y\}$
**[Initialization:]**
$X_0 = \{\emptyset\}; k = 0$
**[Termination:]**
Stop when the criterion does not increase $J(X_k) \approx J(X_{k-1})$[a]
**Step 1 (Inclusion)**
$x^+ = argmax_{x \in Y - X_k} J(X_k \cup x)$
$X_{k+1} = X_k \cup x^+, k = k + 1$
**Step 2 (Conditional exclusion)**
$x^- = argmax_{x \in X_k} J(X_k - x)$
if $J(X_k - x^-) > J(X_{k-1})$ then
$X_{k+1} = X_k - x^-, k = k + 1$
go to **Step 2**
else
go to **Step 1**

[a]We apply the Fast Quadratic Mutual Information *MI*, see Appendix A.

based on embedding dichotomizers in a problem-dependent ECOC design by considering subsets of the original set of classes. In particular, difficult problems where the given base classifier is not flexible enough to distinguish the classes benefit from the subclass strategy. SFFS based on maximizing the MI is used to generate subgroups of problems that are split until the desired performance is achieved. The final set of binary problems is embedded in a problem-dependent ECOC configuration and updated with the new set of subclasses. The experimental results over a set of UCI data sets and on real multiclass traffic sign categorization problems show the utility of the present methodology. It improves the performance of different base classifiers over the state-of-the-art ECOC configurations. This shows that the proposed splitting procedure yields a better performance when the class overlap or the distribution of the training objects conceal the decision boundaries for the base classifier. The results are even more significant when one has a sufficiently large training size.

## APPENDIX A
## SEQUENTIAL FORWARD FLOATING SEARCH (*SFFS*)

The *SFFS* process in Table 7 begins with an empty set $X_0$ and is filled, while the search criterion applied to the new set increases. The most significant item with respect to $X_k$ is added at each inclusion step. In the conditional exclusion step, the worst item is removed if the criterion keeps increasing. $Y$ is our set of classes to be partitioned. Our discriminability criterion is the *MI*. Our goal is to maximize the *MI* between the data in the sets and the class labels created for each subset [21].

## APPENDIX B
## FAST QUADRATIC MUTUAL INFORMATION (*MI*)

Let $\mathbf{x}$ and $\mathbf{y}$ represent two random variables and let $p(\mathbf{x})$ and $p(\mathbf{y})$ be their respective probability density functions. The MI measures the dependence between both variables and is defined as follows:

$$I(\mathbf{x}, \mathbf{y}) = \int\int p(\mathbf{x}, \mathbf{y}) \log \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})p(\mathbf{y})} d\mathbf{x} d\mathbf{y}. \quad (4)$$

Observe that MI is zero if $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$. It is important to note that (4) can be seen as a Kullback-Leiber divergence, defined in the following way:

$$K(f, g) = \int f(y) \log \frac{f(y)}{g(y)} dy, \quad (5)$$

where $f(y)$ is replaced with $p(\mathbf{x}, \mathbf{y})$ and $g(y)$ with $p(\mathbf{x})p(\mathbf{y})$.

Alternatively, Kapur and Kesavan [14] argued that if our goal is to find a distribution that maximizes or minimizes the divergence, several axioms can be relaxed, and the resulting divergence measure is related to $D(f, g) = \int (f(y) - g(y))^2 dy$. As a result, it was proved that maximizing $K(f, g)$ is equivalent to maximizing $D(f, g)$. Therefore, we can define the quadratic MI as follows:

$$I_Q(\mathbf{x}, \mathbf{y}) = \int\int (p(\mathbf{x}, \mathbf{y}) - p(\mathbf{x})p(\mathbf{y}))^2 d\mathbf{x} d\mathbf{y}. \quad (6)$$

The estimation of the density functions of $I_Q$ can be done using the Parzen window estimator. In that case, when combined with Gaussian kernels, we can use the following property: Let $N(y, \Sigma)$ be a $d$-dimensional Gaussian function; it can be shown that

$$\int N(y - a_1, \Sigma_1) N(y - a_2, \Sigma_2) dy = N(a_1 - a_2, \Sigma_1 + \Sigma_2). \quad (7)$$

Observe that the use of this property avoids the computation of one integral function.

In particular, we compute the MI between the random variable of the features $\mathbf{x}$ and the discrete random variable associated to the class labels created for a given partition $(\mathbf{d})$. The notation for the practical implementation of $I_Q$ is given as follows: Assume that we have $N$ samples in the whole data set; $J_p$ are the samples of the class $p$; $N$ stands for the number of classes; $x_l$ stands for the $l$th feature vector of the data set, and $x_{pk}$ is the $k$th feature vector of the set in class $p$. Then, $p(\mathbf{d})$ and $p(\mathbf{x}|\mathbf{d})$ can be written as

$$p(\mathbf{d} = p) = \frac{J_p}{N},$$

$$p(\mathbf{x}|\mathbf{d} = p) = \frac{1}{J_p} \sum_{j=1}^{J_p} N(x - x_{pj}, \sigma^2 I),$$

$$p(\mathbf{x}) = \frac{1}{N} \sum_{j=1}^{N} N(x - x_j, \sigma^2 I).$$

Expanding (6) and using a Parzen estimate with a symmetrical kernel with width $\sigma$, we obtain the following equation:

$$I_Q(\mathbf{x}, \mathbf{d}) = V_{IN} + V_{ALL} - 2V_{BTW}, \quad (8)$$

TABLE 8
Label Loss-Weighted Subclass Decoding Algorithm

Given a coding matrix $M$,
**Step 1** Calculate the matrix of hypothesis $H$:

$$H(i,j) = \frac{1}{|J_i|}\sum_{k=1}^{|J_i|}\gamma(h_j(J_i^k), i, j) \quad (9)$$

$$\text{based on} \quad \gamma(x_j, i, j) = \begin{cases} 1, & \text{if} \quad x_j = M(i,j) \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

**Step 2** Normalize $H$ so that $\sum_{j=1}^{n} M_W(i,j) = 1, \forall i = 1, ..., N$:

$$M_W(i,j) = \frac{H(i,j)}{\sum_{j=1}^{n} H(i,j)}, \forall i \in [1, ..., N], \quad \forall j \in [1, ..., n] \quad (11)$$

**Step 3** Given a test input $\wp$, decode based on:

$$d(x,i) = \sum_{j=1}^{n} -M(i,j) \cdot h(x,j) \cdot M_W(i,j) \quad (12)$$

**Step 4** Obtain the final class $c_x$ for sample $x$ by:

$$C_{i^*} : (i^* j^*) = \underset{(i'j')}{\operatorname{argmin}}\, d(x, (i'j')), \quad C_{(i'j')} \in C' \quad (13)$$

TABLE 9
UCI Repository Experiments for Discrete Adaboost

| Problem | one-versus-one | one-versus-all | dense | sparse | DECOC | Sub-class ECOC |
|---|---|---|---|---|---|---|
| Balance | 78.3(4.9) 3×3 | 76.5(5.9) 3×3 | 80.9(6.2) 3×3 | 78.8(4.1) 3×3 | 78.3(4.9) 3×2 | 79.0(5.0) 3.8×3.2 |
| Wine | 93.7(1.3) 3×3 | 96.05(1.2) 3×3 | 96.0(1.2) 3×3 | 96.0(1.2) 3×3 | 96.0(1.2) 3×2 | 96.0(1.2) 3×2 |
| Thyroid | 92.1(2.7) 3×3 | 92.1(2.7) 3×3 | 92.1(2.7) 3×3 | 92.1(2.7) 3×3 | 92.1(2.7) 3×2 | 92.1(2.7) 3×2 |
| Vowel | 59.0(2.8) 11×55 | 45.6(3.4) 11×11 | 29.7(2.1) 11×11 | 45.1(2.4) 11×11 | 66.7(2.6) 11×10 | 67.4(2.1) 12.1×11.2 |
| Ecoli | 77.8(1.7) 8×28 | 75.5(1.9) 8×8 | 79.7(1.7) 8×8 | 53.9(1.6) 8×8 | 77.0(1.5) 8×7 | 78.2(1.9) 9×10.1 |
| Iris | 93.3(2.2) 3×3 | 93.3(2.2) 3×3 | 93.3(2.2) 3×3 | 93.3(2.2) 3×3 | 93.3(2.2) 3×2 | 93.3(2.2) 3×2 |
| Yeast | 49.1(1.5) 10×45 | 41.3(1.2) 10×10 | 46.9(1.8) 10×10 | 39.5(1.4) 10×10 | 51.8(1.3) 10×9 | 52.0(1.2) 10.7×12.3 |
| Glass | 67.1(3.1) 7×21 | 60.2(3.7) 7×7 | 52.7(4.0) 7×7 | 59.2(3.7) 7×7 | 66.5(2.8) 7×6 | 66.8(2.6) 8×8.1 |

where

$$V_{IN} = \sum \int p(\mathbf{x},\mathbf{d})^2 dx = \frac{1}{N^2}\sum_{p=1}^{N}\sum_{l=1}^{J_p}\sum_{k=1}^{J_p} N(x_{pl} - x_{pk}, 2\sigma^2 I),$$

$$V_{ALL} = \sum \int p(\mathbf{x})^2 p(\mathbf{d})^2 dx,$$

$$V_{ALL} = \frac{1}{N^2}\sum\nolimits_{p=1}^{N}\left(\frac{J_p}{N}\right)^2 \sum_{l=1}^{N}\sum_{k=1}^{N} N(x_l - x_k, 2\sigma^2 I),$$

$$V_{BTW} = \sum \int p(\mathbf{x},\mathbf{d}) p(\mathbf{x}) p(\mathbf{d}) dx,$$

$$V_{BTW} = \frac{1}{N^2}\sum_{p=1}^{N}\frac{J_p}{N}\sum_{l=1}^{N}\sum_{k=1}^{J_p} N(x_l - x_{pk}, 2\sigma^2 I).$$

In practical applications, $\sigma$ is usually set to the half of the maximum distance between samples, as proposed by Torkkola [23].

## APPENDIX C
## SUBCLASS ECOC DECODING ALGORITHM

Table 8 summarizes the decoding strategy applied for the subclass ECOC. The main objective is to find a weighting matrix $M_W$ that weights a given decoding measure to adjust the decisions of the classifiers. To obtain the weighting matrix $M_W$, we assign to each position $(i,j)$ of the matrix of hypothesis $H$ a continuous value that corresponds to the accuracy of the dichotomy $h_j$, classifying the samples of class $i$ (9). We make $H$ to have zero probability at those positions corresponding to unconsidered classes (10) since these positions do not have representative information. The next step is to normalize each row of the matrix $H$ so that $M_W$ can be considered as a discrete probability density function (11). This step assumes that the probability of considering each class for the final classification is the same (independent of number of zero symbols) in the case of not having a priori information $(P(c_1) = \ldots = P(c_N))$ [8]. The final classification decision is obtained by assigning the original class $C_{i^*}$ of the subclass $C_{(i^*j^*)}$ that attains the minimum decoding measure $d$.

TABLE 10
UCI Repository Experiments for *NMC*

| Problem | one-versus-one | one-versus-all | dense | sparse | DECOC | Sub-class ECOC |
|---|---|---|---|---|---|---|
| Balance | 78.1(5.2) 3×3 | 77.7(4.7) 3×3 | 77.6(5.1) 3×3 | 70.3(5.7) 3×3 | 78.1(4.2) 3×2 | 79.3(4.8) 6.4×9.3 |
| Wine | 71.3(4.9) 3×3 | 67.9(4.7) 3×3 | 66.8(3.7) 3×3 | 69.1(3.9) 3×3 | 92.6(3.3) 3×2 | 96.7(3.2) 7.2×11.5 |
| Thyroid | 88.9(2.0) 3×3 | 81.8(2.1) 3×3 | 83.1(2.6) 3×3 | 80.0(2.3) 3×3 | 83.5(2.1) 3×2 | 93.7(2.2) 5.1×6.4 |
| Vowel | 58.1(2.3) 11×55 | 47.2(4.1) 11×11 | 38.6(5.3) 11×11 | 35.7(3.6) 11×11 | 54.5(3.6) 11×10 | 63.9(3.7) 22.3×24.7 |
| Ecoli | 82.5(2.0) 8×28 | 64.9(1.8) 8×8 | 73.4(2.2) 8×8 | 65.0(2.5) 8×8 | 83.1(2.5) 8×7 | 84.5(2.8) 14.3×26.8 |
| Iris | 92.6(1.6) 3×3 | 78.6(2.2) 3×3 | 78.3(2.4) 3×3 | 59.3(3.6) 3×3 | 91.1(3.0) 3×2 | 94.0(2.8) 6.3×9.5 |
| Yeast | 52.0(2.4) 10×45 | 48.8(2.3) 10×10 | 44.0(2.7) 10×10 | 48.7(2.7) 10×10 | 49.0(3.1) 10×9 | 49.2(2.8) 13.2×14.4 |
| Glass | 41.9(4.8) 7×21 | 25.1(4.4) 7×7 | 30.8(4.8) 7×7 | 35.6(5.2) 7×7 | 48.8(4.0) 7×6 | 66.9(3.8) 16.6×29.4 |

TABLE 11
UCI Repository Experiments for *FLDA*

| Problem | one-versus-one | one-versus-all | dense | sparse | DECOC | Sub-class ECOC |
|---|---|---|---|---|---|---|
| Balance | 80.7(4.8) 3×3 | 80.7(4.8) 3×3 | 80.7(4.8) 3×3 | 80.7(4.8) 3×3 | 80.7(4.8) 3×2 | 80.7(4.8) 3×2 |
| Wine | 96.5(2.9) 3×3 | 96.5(2.9) 3×3 | 96.4(3.0) 3×3 | 96.5(2.9) 3×3 | 96.7(2.7) 3×2 | 96.7(2.7) 3×2 |
| Thyroid | 94.8(3.4) 3×3 | 88.8(4.0) 3×3 | 90.6(4.3) 3×3 | 88.8(4.5) 3×3 | 86.0(3.7) 3×2 | 93.8(3.6) 5.3×6.4 |
| Vowel | 70.3(3.8) 11×55 | 45.1(4.7) 11×11 | 46.5(4.3) 11×11 | 41.6(3.1) 11×11 | 67.5(2.8) 11×10 | 74.2(3.2) 19.9×32.8 |
| Ecoli | 85.2(3.5) 8×28 | 77.5(3.4) 8×8 | 78.3(3.9) 8×8 | 49.8(3.8) 8×8 | 85.2(3.4) 8×7 | 85.2(3.4) 8×7 |
| Iris | 98.0(2.0) 3×3 | 91.3(3.3) 3×3 | 93.3(3.4) 3×3 | 66.6(1.3) 3×3 | 97.7(2.1) 3×2 | 97.7(2.1) 3×2 |
| Yeast | 51.8(2.8) 10×45 | 30.0(3.3) 10×10 | 46.0(3.6) 10×10 | 45.5(3.4) 10×10 | 51.3(2.1) 10×9 | 51.3(2.1) 10×9 |
| Glass | 60.0(4.6) 7×21 | 46.8(4.9) 7×7 | 51.8(3.9) 7×7 | 53.3(4.2) 7×7 | 61.1(3.2) 7×6 | 63.0(3.7) 8.8×10.5 |

TABLE 12
UCI Repository Experiments for Linear *SVM*

| Problem | one-versus-one | one-versus-all | dense | sparse | DECOC | Sub-class ECOC |
|---|---|---|---|---|---|---|
| Balance | 84.6(3.2) 3×3 | 85.5(3.1) 3×3 | 85.5(3.1) 3×3 | 85.5(3.1) 3×3 | 85.5(3.1) 3×2 | 85.5(3.1) 3×2 |
| Wine | 93.7(1.7) 3×3 | 93.2(1.6) 3×3 | 93.2(1.6) 3×3 | 93.2(1.6) 3×3 | 95.5(1.4) 3×2 | 98.0(1.6) 8.7×11.3 |
| Thyroid | 94.3(2.1) 3×3 | 94.3(2.1) 3×3 | 94.3(2.1) 3×3 | 94.3(2.1) 3×3 | 94.3(2.1) 3×2 | 94.3(2.1) 3×2 |
| Vowel | 65.9(3.6) 11×55 | 32.8(2.1) 11×11 | 31.1(3.0) 11×11 | 35.7(2.5) 11×11 | 58.8(3.4) 11×10 | 68.7(2.1) 16.3×21.8 |
| Ecoli | 78.6(2.4) 8×28 | 68.8(3.4) 8×8 | 71.5(2.7) 8×8 | 68.3(3.8) 8×8 | 79.2(2.4) 8×7 | 81.5(2.4) 11.5×14.7 |
| Iris | 97.3(1.0) 3×3 | 97.3(1.0) 3×3 | 97.3(1.0) 3×3 | 97.3(1.07) 3×3 | 97.3(1.0) 3×2 | 97.3(1.0) 3×2 |
| Yeast | 51.1(4.2) 10×45 | 17.0(3.4) 10×10 | 40.5(1.2) 10×10 | 34.1(1.7) 10×10 | 51.1(2.6) 10×9 | 53.5(2.5) 17.8×26.1 |
| Glass | 55.5(3.2) 7×21 | 41.3(6.4) 7×7 | 37.7(2.8) 7×7 | 44.3(2.1) 7×7 | 63.8(3.1) 7×6 | 66.9(2.8) 10.7×13.2 |

## APPENDIX D
## TABLES OF PERFORMANCES OF THE UCI EXPERIMENTS

Tables 9, 10, 11, 12, and 13 show the performance of the UCI experiments for the different base classifiers. Each position of the table contains the performance obtained applying 10-fold

TABLE 13
UCI Repository Experiments for *RBF SVM*

| Problem | one-versus-one | one-versus-all | dense | sparse | DECOC | Sub-class ECOC |
|---|---|---|---|---|---|---|
| Balance | 80.4(3.2)<br>3×3 | 69.2(3.9)<br>3×3 | 71.0(3.5)<br>3×3 | 69.2(2.9)<br>3×3 | 83.0(3.8)<br>3×2 | 83.0(3.8)<br>3×2 |
| Wine | 39.9(0.8)<br>3×3 | 33.1(1.0)<br>3×3 | 33.1(1.3)<br>3×3 | 33.1(1.0)<br>3×3 | 35.8(1.1)<br>3×2 | 90.8(3.1)<br>4.5×6.2 |
| Thyroid | 90.7(1.0)<br>3×3 | 89.8(1.6)<br>3×3 | 90.7(1.0)<br>3×3 | 90.7(1.0)<br>3×3 | 91.7(1.2)<br>3×2 | 93.7(1.3)<br>3.6×4.1 |
| Vowel | 82.5(2.1)<br>11×55 | 52.5(2.2)<br>11×11 | 72.2(3.6)<br>11×11 | 47.9(3.8)<br>11×11 | 73.6(3.2)<br>11×10 | 75.9(2.4)<br>12.4×13.5 |
| Ecoli | 80.1(3.2)<br>8×28 | 78.7(4.4)<br>8×8 | 84.2(2.8)<br>8×8 | 75.2(3.1)<br>8×8 | 82.2(3.2)<br>8×7 | 84.2(3.8)<br>9.8×10.3 |
| Iris | 96.0(2.8)<br>3×3 | 96.0(2.8)<br>3×3 | 96.0(2.8)<br>3×3 | 96.0(2.8)<br>3×3 | 96.0(2.8)<br>3×2 | 96.0(2.8)<br>3×2 |
| Yeast | 52.1(2.5)<br>10×45 | 45.5(3.2)<br>10×10 | 52.4(2.9)<br>10×10 | 46.7(2.7)<br>10×10 | 51.6(2.1)<br>10×9 | 53.2(3.2)<br>12.1×14.7 |
| Glass | 64.7(3.5)<br>7×21 | 51.0(3.4)<br>7×7 | 64.7(3.4)<br>7×7 | 37.4(2.2)<br>7×7 | 63.9(4.2)<br>7×6 | 66.1(3.2)<br>8.5×10 |

cross validation and the confidence interval at 95 percent. The mean number of classes (or subclasses) and the mean number of binary problems are shown below each performance.

## ACKNOWLEDGMENTS

This research is/was supported in part by the projects TIN2006-15308-C02, FIS PI061290, Dutch Technology Foundation STW, Applied Science Division of NWO, and the technology program of the Dutch Ministry of Economic Affairs.

## REFERENCES

[1] E. Allwein, R. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *J. Machine Learning Research,* vol. 1, pp. 113-141, 2002.
[2] A. Asuncion and D. Newman, *UCI Machine Learning Repository.* School of Information and Computer Sciences, Univ. of California, Irvine, 2007.
[3] J. Casacuberta, J. Miranda, M. Pla, S. Sanchez, A. Serra, and J. Talaya, "On the Accuracy and Performance of the Geomobil System," *Proc. 20th Congress Int'l Soc. for Photogrammetry and Remote Sensing*, 2004.
[4] K. Crammer and Y. Singer, "On the Learnability and Design of Output Codes for Multi-Class Problems," *Machine Learning,* vol. 47, pp. 201-233, 2002.
[5] H. Daume and D. Marcu, "A Bayesian Model for Supervised Clustering with the Dirichlet Process Prior," *J. Machine Learning Research,* pp. 1551-1577, 2005.
[6] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *J. Machine Learning Research,* pp. 1-30, 2006.
[7] T. Dietterich and G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *J. Artificial Intelligence Research,* vol. 2, pp. 263-282, 1995.
[8] S. Escalera, O. Pujol, and P. Radeva, "Loss-Weighted Decoding for Error-Correcting Output Codes," *Proc. Int'l Conf. Computer Vision Theory and Applications,* vol. 2, pp. 117-122, 2008.
[9] S. Escalera, O. Pujol, and P. Radeva, "Boosted Landmarks of Contextual Descriptors and Forest-ECOC: A Novel Framework to Detect and Classify Objects in Clutter Scenes," *Pattern Recognition Letters,* vol. 28, no. 13, pp. 1759-1768, 2007.
[10] T.N. Faculty of Applied Physics, Delft Univ. of Technology, http://www.prtools.org/, 2008.
[11] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics,* vol. 38, pp. 337-374, 1998.
[12] R. Ghani, "Combining Labeled and Unlabeled Data for Text Classification with a Large Number of Categories," *Proc. IEEE Int'l Conf. Data Mining,* pp. 597-598, 2001.
[13] T. Hastie and R. Tibshirani, "Classification by Pairwise Grouping," *Proc. Neural Information Processing Systems Conf.,* vol. 26, pp. 451-471, 1998.
[14] J. Kapur and H. Kesavan, *Entropy Optimization Principles with Applications.* Academic Press, 1992.
[15] J. Kittler, R. Ghaderi, T. Windeatt, and J. Matas, "Face Verification Using Error Correcting Output Codes," *Proc. Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 755-760, 2001.
[16] E.B. Kong and T.G. Dietterich, "Error-Correcting Output Coding Corrects Bias and Variance," *Proc. 12th Int'l Conf. Machine Learning,* pp. 313-321, 1995.
[17] N.J. Nilsson, *Learning Machines.* McGraw-Hill, 1965.
[18] OSU-SVM-TOOLBOX, http://svm.sourceforge.net/docs/3.00/api/, 2008.
[19] P. Pudil, F. Ferri, J. Novovicova, and J. Kittler, "Floating Search Methods for Feature Selection with Nonmonotonic Criterion Functions," *Proc. Int'l Conf. Pattern Recognition,* pp. 279-283, 1994.
[20] O. Pujol, S. Escalera, and P. Radeva, "An Incremental Node Embedding Technique for Error Correcting Output Codes," *Pattern Recognition,* submitted for publication.
[21] O. Pujol, P. Radeva, and J. Vitrià, "Discriminant ECOC: A Heuristic Method for Application Dependent Design of Error Correcting Output Codes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 6, pp. 1001-1007, June 2006.
[22] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.
[23] J. Shi and J. Malik, "Feature Extraction by Non-Parametric Mutual Information Maximization," *J. Machine Learning Research,* pp. 1415-1438, 2003.
[24] W. Utschick and W. Weichselberger, "Stochastic Organization of Output Codes in Multiclass Learning Problems," *Neural Computation,* vol. 13, no. 5, pp. 1065-1102, 2001.
[25] T. Windeatt and G. Ardeshir, "Boosted ECOC Ensembles for Face Recognition," *Proc. Int'l Conf. Visual Information Eng.,* pp. 165-168, 2003.
[26] Q. Zgu, "Minimum Cross-Entropy Approximation for Modeling of Highly Intertwining Data Sets at Subclass Levels," *J. Intelligent Information Systems,* pp. 139-152, 1998.
[27] J. Zhou and C. Suen, "Unconstrained Numeral Pair Recognition Using Enhanced Error Correcting Output Coding: A Holistic Approach," *Proc. Eighth Int'l Conf. Document Analysis and Recognition,* vol. 1, pp. 484-488, 2005.
[28] M. Zhu and A.M. Martinez, "Subclass Discriminant Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 8, pp. 1274-1286, Aug. 2006.

**Sergio Escalera** received the BS and MS degrees from Universitat Autònoma de Barcelona in 2003 and 2005, respectively. He is currently working toward the PhD degree in computer science, under the supervision of Dr. Petia Radeva and Dr. Oriol Pujol. He is a collaborator professor at the Universitat de Barcelona. His research interests include machine learning, statistical pattern recognition, and visual object recognition.

**David M.J. Tax** received the MSc degree in physics from the University of Nijmegen, The Netherlands, in 1996 and the PhD degree from the Delft University of Technology, The Netherlands, in 2001, with a thesis on the problem of one-class classification or novelty detection. He is currently working on a European Community Marie Curie Fellowship called "One-class classification" in the Intelligent Data Analysis Group, Fraunhofer FIRST, Berlin, in close collaboration with the Delft University of Technology. His research interests include pattern recognition and machine learning, with a focus on outlier detection and novelty detection, the feature selection for and the evaluation of one-class classifiers.

**Oriol Pujol** received the PhD degree in computer science from the Universitat Autònoma de Barcelona in 2004. He is currently a lecturer at the Universitat de Barcelona. His main research interest includes statistical machine learning techniques for object recognition and medical imaging analysis.

**Petia Radeva** received the PhD degree in development of physics-based models applied to image analysis from the Universitat Autònoma de Barcelona (UAB). She is currently an associate professor in the Computer Science Department, UAB. She joined the Applied Mathematics and Analysis Department, Universitat de Barcelona in 2007. Her present research interest is concentrated on the development of physics-based and statistical approaches for object recognition, medical image analysis, and industrial vision. She is a member of the IEEE.

**Robert P.W. Duin** studied applied physics at Delft University of Technology in The Netherlands. In 1978, he received the PhD degree for a thesis on the accuracy of statistical pattern recognizers. In his research, he included various aspects of the automatic interpretation of measurements, learning systems, and classifiers. From 1980 to 1990, he studied and developed hardware architectures and software configurations for interactive image analysis. He is currently an associate professor in the Faculty of Applied Sciences, Delft University of Technology. His present research interest is in the design and evaluation of learning algorithms for pattern recognition applications. This includes, in particular, neural network classifiers, support vector classifiers, and classifier combining strategies. Recently, he began studying the possibilities of relational methods for pattern recognition.