

# Sub-class Error-Correcting Output Codes

Sergio Escalera, Oriol Pujol and Petia Radeva

Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Spain.  
Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona,  
Gran Via 585, 08007, Barcelona, Spain.

**Abstract.** A common way to model multi-class classification problems is by means of Error-Correcting Output Codes (ECOC). One of the main requirements of the ECOC design is that the base classifier is capable of splitting each sub-group of classes from each binary problem. In this paper, we present a novel strategy to model multi-class classification problems using sub-class information in the ECOC framework. Complex problems are solved by splitting the original set of classes into sub-classes, and embedding the binary problems in a problem-dependent ECOC design. Experimental results over a set of UCI data sets and on a real multi-class traffic sign categorization problem 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.

## 1 Introduction

In the literature, one can find several powerful binary classifiers. However, when one needs to deal with multi-class 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 were born as a general framework to combine binary problems to address the multi-class problem [3].

The ECOC technique can be broken down into two distinct stages: encoding and decoding. Given a set of classes, the coding stage designs a codeword<sup>1</sup> for each class based on different binary problems, that are combined in a coding matrix  $M$ . The decoding stage makes a classification decision for a given test sample based on the value of the output code.

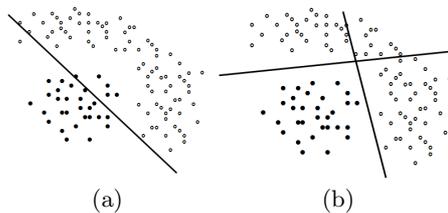
It was when Allwein et al. [8] introduced a third symbol (the zero symbol) in the coding process when the coding step received special attention. This symbol increases the number of partitions of classes to be considered in a ternary ECOC framework by allowing some classes to be ignored. Then, the ternary coding matrix becomes  $M \in \{-1, 0, 1\}^{N \times n}$ , for  $N$  number of classes and  $n$  number of binary problems. In this case, the symbol zero means that a particular class is not considered by a certain binary classifier. Recently, new improvements in the

---

<sup>1</sup> The codeword 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.

ternary ECOC coding demonstrate the suitability of the ECOC methodology to deal with multi-class classification problems [6][7]. 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 matrix  $M$  focusing on the binary problems that better fit the decision boundaries of a given data set. 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 [1], and expensive parameter optimizations. Look at the example of fig. 1(a). 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. 1(b), one of the previous classes has been split into two sub-sets, that we call *sub-classes*. Then, the original problem is solved using two linear classifiers, and the two new sub-classes have the same original class label. Some studies in the literature tried to form sub-classes using the labels information, which is called Supervised Clustering [10]. 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 [11], the authors use the class labels to form the sub-classes that improve the performance of particular Discriminant Analysis algorithms.

In this paper, we present a problem-dependent ECOC design where classes are partitioned into sub-classes using a clustering approach for the cases that the base classifier is not capable to distinguish the classes. Sub-groups of problems are split into more simple ones until the base classifier is able to learn the original problem. In this way, multi-class problems which can not be modelled by using the original set of classes are modelled without the need of using more complex classifiers. The final ECOC design is obtained by combining the sub-problems. The novel Sub-class ECOC design is compared with the state-of-art ECOC designs over a set of UCI Machine Learning Repository data sets and on a real multi-class traffic sign categorization problem using different base classifiers. The results show that in most cases the sub-class strategy is able to obtain significant performance improvements.



**Fig. 1.** (a) Decision boundary of a linear classifier of a 2-class problem. (b) Decision boundaries of a linear classifier splitting the problem of (a) into two more simple tasks.

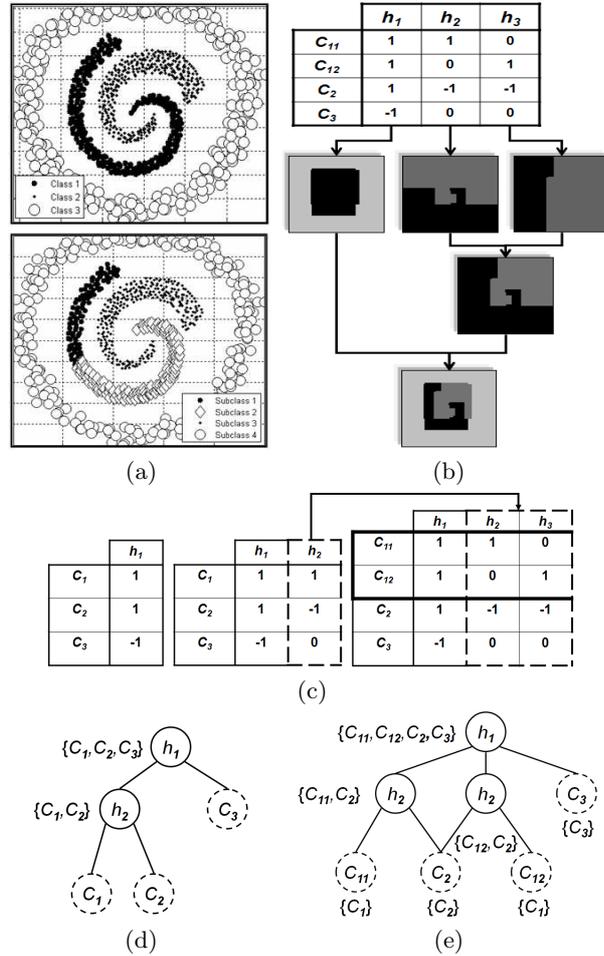
## 2 Problem-dependent ECOC Sub-class

From an initial set of classes  $C$  of a given multi-class problem, the objective of the Sub-class 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 sub-sets if necessary.

Recently, the authors of [6] proposed a ternary problem-dependent design of ECOC, called DECOC. 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 mutual information ( $MI$ ) between the data and their respective class labels. Look at the 3-class problem shown on the top of fig. 2(a). The DECOC algorithm considers the whole set of classes to split it into two sub-sets of classes  $\varphi^+$  and  $\varphi^-$  maximizing the  $MI$  criterion on a sequential forward floating search procedure ( $SFFS$ ). In the example, the first sub-sets found correspond to  $\varphi^+ = \{C_1, C_2\}$  and  $\varphi^- = \{C_3\}$ . 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. 2(d). The procedure is repeated until all classes are split into separate sub-sets  $\varphi$ . In the example, the second classifier is trained to split the sub-sets of classes  $\varphi^+ = C_1$  from  $\varphi^- = C_2$  because the classes  $C_1$  and  $C_2$  were still contained in a single sub-set after the first step. This second classifier is codified by the node  $h_2$  of fig. 2(d). 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. 2(c)).

In our case, sequential forward floating search ( $SFFS$ ) is also applied to look for the sub-sets  $\varphi^+$  and  $\varphi^-$  that maximizes the mutual information between the data and their respective class labels [6]. The  $SFFS$  algorithm used is the one proposed in [12], and the implementation details of the fast quadratic mutual information can be found in [6]. To illustrate our procedure, let us return to the example of the top of fig. 2(a). On the first iteration of the sub-class ECOC algorithm,  $SFFS$  finds the sub-set  $\varphi^+ = \{C_1, C_2\}$  against  $\varphi^- = \{C_3\}$ . The encoding of this problem is shown in the first matrix of fig. 2(c). 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 fig. 2(b). One can see that the base classifier finds a good solution for this first problem.

Then, the second classifier is trained to split  $\varphi^+ = C_1$  against  $\varphi^- = C_2$ , and its performance is computed. To separate the current sub-sets is not a trivial problem, and the classification performance is poor. Therefore, our procedure tries to split the data  $J_{\varphi^+}$  and  $J_{\varphi^-}$  from the current sub-sets  $\varphi^+$  and  $\varphi^-$  into more simple sub-sets. Applying a splitting criteria  $SC$  over the two sub-sets, two clusters are found for  $\varphi^+ = C_1$  and for  $\varphi^- = C_2$ . We select the split that maximizes the distance between the means of the clusters. And then, the original encoding of the problem  $C_1$  vs  $C_2$  is transformed to two more simple problems  $\{C_{11}\}$



**Fig. 2.** (a) Top: Original 3-class problem. Bottom: 4 sub-classes found. (b) Sub-class ECOC encoding using the four sub-classes using Discrete Adaboost with 40 runs of Decision Stumps. (c) Learning evolution of the sub-class matrix  $M$ . (d) Original tree structure without applying sub-class. (e) New tree-based configuration using sub-classes.

against  $\{C_2\}$  and  $\{C_{12}\}$  against  $\{C_2\}$ . It implies that the class  $C_1$  is split into two sub-classes (look at the bottom of fig. 2(a)), and the original 3-class problem  $C = \{C_1, C_2, C_3\}$  becomes the 4-sub-class 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, previous dichotomizers that take this class into account need to be updated. The dichotomizer  $h_1$  considers the sub-sets  $\wp_1^+ = \{C_1, C_2\}$  and  $\wp_1^- = \{C_3\}$ . Then, those positions containing class  $C_1$  are replaced with  $C_{11}$  and  $C_{12}$ . Now, the original tree encoding of the DECOC design shown in fig. 2(d) can be represented by the tree structure of fig. 2(e), where the original class associated to each sub-class is shown in the leaves.

## 2.1 Sub-class algorithm

The encoding algorithm is shown in table 1. Given a  $N$ -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  $k$  (**Step 1**), the first element of  $L$  is assigned to  $S_k$  in the first step of the algorithm, and the optimal binary partition  $BP$  of  $S_k$  is found (**Step 2**).

**Table 1.** Problem-dependent Sub-class ECOC algorithm.

<p><b>Inputs:</b> <math>J, C, \theta = \{\theta_{size}, \theta_{perf}, \theta_{impr}\}</math> // Thresholds for the number of samples, performance, and improvement between iterations</p> <p><b>Outputs:</b> <math>C', J', \varphi', M</math></p> <p><b>[Initialization:]</b></p> <p>Create the trivial partition <math>\{\varphi_0^+, \varphi_0^-\}</math> of the set of classes <math>\{C_i\}</math>: <math>\{\varphi_0^+, \varphi_0^-\} = \{\{\emptyset\}, \{C_1, C_2, \dots, C_N\}\}</math></p> <p><math>L_0 = \{\varphi_0^-\}; J' = J; C' = C; \varphi' = \emptyset; M = \emptyset; k = 1</math></p> <p><b>Step 1</b> <math>S_k</math> is the first element of <math>L_{k-1}</math></p> <p><math>L'_k = L_{k-1} \setminus \{S_k\}</math></p> <p><b>Step 2</b> Find the optimal binary partition <math>BP(S_k)</math>:</p> <p><math>\{\varphi_k^+, \varphi_k^-\} = \underset{BP(S_k)}{\operatorname{argmax}} (I(\mathbf{x}, d(BP(S_k))))</math></p> <p>where <math>I</math> is the mutual information criterion, <math>\mathbf{x}</math> is the random variable associated to the features and <math>d</math> is the discrete random variable of the dichotomy labels<sup>a</sup>, defined in the following terms,</p> $d = d(\mathbf{x}, BP(S_k)) = \begin{cases} 1 & \text{if } \mathbf{x} \in C_i   C_i \in \varphi_k^+ \\ -1 & \text{if } \mathbf{x} \in C_i   C_i \in \varphi_k^- \end{cases}$ <p><b>Step 3</b> // Look for sub-classes</p> <p><math>\{C', J', \varphi'\} = \operatorname{SPLIT}(J_{p_k^+}, J_{p_k^-}, C', J', J, \varphi', \theta)</math><sup>b</sup></p> <p><b>Step 4</b> <math>L_k = \{L'_k \cup \varphi_k^i\}</math> if <math> \varphi_k^i  &gt; 1 \forall i \in \{+, -\}</math></p> <p><b>Step 5</b> If <math> L_k  \neq 0</math></p> <p><math>k = k + 1</math> <b>go to Step 1</b></p> <p><b>Step 6</b> Codify the coding matrix <math>M</math> using each partition <math>\{\varphi_i^+, \varphi_i^-\}</math> of <math>\varphi'</math>, <math>i \in [1, \dots,  \varphi' ]</math> and each class <math>C_r \in \varphi_i = \{\varphi_i^+ \cup \varphi_i^-\}</math> as follows:</p> $M(C_r, i) = \begin{cases} 0 & \text{if } C_r \notin \varphi_i \\ +1 & \text{if } C_r \in \varphi_i^+ \\ -1 & \text{if } C_r \in \varphi_i^- \end{cases} \quad (1)$
<p><sup>a</sup> Use <i>SFFS</i> of [12] as the maximization procedure and <i>MI</i> of [6] to estimate <math>I</math></p> <p><sup>b</sup> Using the splitting algorithm of table 2.</p>

At **Step 3** of the algorithm, the splitting criteria  $SC$  takes as input a data set  $J_{\varphi^+}$  or  $J_{\varphi^-}$  from a sub-set  $\varphi^+$  or  $\varphi^-$ , and splits it into two sub-sets  $J_{\varphi^+}^+$  and  $J_{\varphi^+}^-$  or  $J_{\varphi^-}^+$  and  $J_{\varphi^-}^-$ . The splitting algorithm is shown in table 2.

When two data sub-sets  $\{J_{\varphi^+}^+, J_{\varphi^+}^-\}$  and  $\{J_{\varphi^-}^+, J_{\varphi^-}^-\}$  are obtained, we select the sub-sets that have the highest distance between the means of each cluster. Suppose that the distance between  $J_{\varphi^+}^+$  and  $J_{\varphi^-}^-$  is larger than between  $J_{\varphi^+}^+$  and  $J_{\varphi^+}^-$ . Then, only  $J_{\varphi^+}$ ,  $J_{\varphi^-}^+$ , and  $J_{\varphi^-}^-$  are used. If the new sub-sets improve the classification performance, new sub-classes are formed, and the process is repeated.

The function *TEST\_PARAMETERS* in table 2 is responsible for testing the constraints based on the parameters  $\{\theta_{size}, \theta_{perf}, \theta_{impr}\}$ . If the constraints are satisfied, the new sub-sets are selected and used to recursively call the splitting function (**Step 3** of the algorithm in table 2). The constraints of the function *TEST\_PARAMETERS* are fixed as: 1) The number of objects in  $J_{\varphi^+}$  has to be

**Table 2.** Sub-class *SPLIT* algorithm.

<p><b>Inputs:</b> <math>J_{\varphi 1}, J_{\varphi 2}, C', J', J, \varphi', \theta</math> // <math>C'</math> is the final set of classes, <math>J'</math> the data for the final set of classes, and <math>\varphi'</math> is the labels for all the partitions of classes of the final set.</p> <p><b>Outputs:</b> <math>C', J', \varphi'</math></p> <p><b>Step 1</b> Split problems:  <math>\{J_{\varphi+}^+, J_{\varphi+}^-\} = SC(J_{\varphi+})^a</math>  <math>\{J_{\varphi-}^+, J_{\varphi-}^-\} = SC(J_{\varphi-})</math></p> <p><b>Step 2</b> Select sub-classes:  if <math> J_{\varphi+}^+, J_{\varphi+}^-  &gt;  J_{\varphi-}^+, J_{\varphi-}^- </math> // find the largest distance between the means of each sub-set.  <math>\{J_{\varphi+}^+, J_{\varphi+}^-\} = \{J_{\varphi+}^+, J_{\varphi-}^-\}; \{J_{\varphi-}^+, J_{\varphi-}^-\} = \{J_{\varphi+}^-, J_{\varphi-}^-\}</math>  else  <math>\{J_{\varphi+}^+, J_{\varphi+}^-\} = \{J_{\varphi-}^+, J_{\varphi+}^-\}; \{J_{\varphi-}^+, J_{\varphi-}^-\} = \{J_{\varphi-}^-, J_{\varphi+}^-\}</math>  end</p> <p><b>Step 3</b> Test parameters to continue splitting:  if <math>TEST\_PARAMETERS(J_{\varphi 1}, J_{\varphi 2}, J_1^1, J_1^2, J_2^1, J_2^2, \theta)</math> // call the function with the new sub-sets  <math>\{C', J', \varphi'\} = SPLIT(J_1^1, J_1^2, C', J', J, \varphi', \theta)</math>  <math>\{C', J', \varphi'\} = SPLIT(J_2^1, J_2^2, C', J', J, \varphi', \theta)</math>  end</p> <p><b>Step 4</b> Save the current partition:  Update the data for the new sub-classes and previous sub-classes if intersections exists <math>J'</math>.  Update the final number of sub-classes <math>C'</math>.  Create <math>\varphi_c = \{\varphi_{c1}, \varphi_{c2}\}</math> the set of labels of the current partition.  Update the labels of the previous partitions <math>\varphi</math>.  Update the set of partitions labels with the new partition <math>\varphi' = \varphi' \cup \varphi_c</math>.</p>
<p><sup>a</sup> <math>SC</math> corresponds to the splitting method of the input data into two main clusters.</p>

larger than  $\theta_{size}$ , 2) The number of objects in  $J_{\varphi-}$  has to be larger than  $\theta_{size}$ , 3) The error  $\xi(h(J_{\varphi-}, J_{\varphi+}))$  obtained from the dichomomizer  $h$  using a particular base classifier applied on the sets  $\{\varphi^+, \varphi^-\}$  has to be larger than  $\theta_{perf}$ , and 4) The sum of the well-classified new objects (based on the confusion matrices) divided by the total number of objects has to be greater than  $1 - \theta_{impr}$ .

$\theta_{size}$  corresponds to the minimum number of object samples that has to be in a sub-set,  $\theta_{perf}$  is the threshold for the performance of the current binary problem, and  $\theta_{impr}$  looks for the performance improvements of the split groups in relation with the previous one.

When a new sub-class is formed, we need to save the information of the current sub-sets  $\{\varphi^+, \varphi^-\}$  and the previous sub-sets affected by the new splitting (**Step 4** of the splitting algorithm). When the final set of binary problems is obtained, its respective set of labels  $\varphi'$  is used to create the coding matrix  $M$  (eq. (1)).

Finally, to decode the new sub-class problem-dependent design of ECOC, we take advantage of the recently proposed Loss-Weighted decoding design [9]. The decoding strategy uses a set of normalized probabilities based on the performance of the base classifier and the ternary ECOC constraints [9].

### 3 Experimental results

In order to evaluate the methodology, we discuss the data, compared methods, experiments, and performance evaluation.

- *Data:* The data used for the experiments consists of eight arbitrary multi-class data sets from the UCI Machine Learning Repository [4] and one real

9-class traffic sign classification problem from the Geomobil project of [5]. The characteristics of the UCI data sets are shown in table 3.

**Table 3.** UCI Machine Learning Repository data sets characteristics.

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

- *Compared methods:* We compare our method with the state-of-the-art ECOC coding designs: one-versus-one, one-versus-all, dense random, sparse random [8], and DECOC [6]. 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 [2], Linear Discriminant Analysis, Linear Support Vector Machines with the regularization parameter  $C$  set to 1 [1], and Support Vector Machines with Radial Basis Function kernel with the default values of the regularization parameter  $C$  and the gamma parameter set to 1 [1]<sup>2</sup>.

- *Experiments:* First, we illustrate the effect of the sub-class algorithm over toy problems. Second, we classify the set of UCI Machine Learning Repository data sets with the ECOC designs and the different base classifiers. Finally, a real multi-class traffic sign recognition problem is evaluated.

- *Performance evaluation:* To evaluate the performance of the different experiments, we apply stratified ten-fold cross-validation and test for the confidence interval at 95% with a two-tailed t-test.

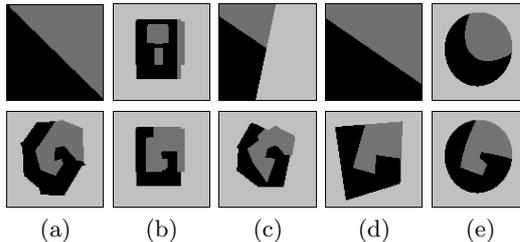
### 3.1 Illustration over toy problems

To show the effect of the Sub-class ECOC strategy for different base classifiers, we used the previous toy problem of the top of fig. 2(a). Using the previously commented base classifiers on the toy problem, the original DECOC strategy with the Loss-Weighted algorithm obtains the decision boundaries shown on the top row of fig. 3. The new learned boundaries are shown on the bottom row of fig. 3 for fixed parameters  $\theta$ . Depending on the flexibility of the base classifier more sub-classes 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.

### 3.2 UCI Machine Learning Repository

Using the UCI data sets of table 3, the five base classifiers, and the six ECOC designs, we have performed a total of 240 ten-fold tests. The set of parameters of the sub-class approach  $\theta = \{\theta_{size}, \theta_{perf}, \theta_{impr}\}$  has been fixed to  $\theta_{size} = \frac{|J|}{50}$  minimum number of objects to apply sub-class (thus, 2% of the samples of

<sup>2</sup> We selected this parameter after a preliminary set of experiments.



**Fig. 3.** Sub-class ECOC without sub-classes (top) and including sub-classes (bottom): for *FLDA* (a), *Discrete Adaboost* (b), *NMC* (c), *Linear SVM* (d), and *RBF SVM* (e).

each particular problem),  $\theta_{perf} = 0$  to split classes if the binary problem does not learn properly the training objects, and  $\theta_{impr} = 0.95$ , that means that the split problems must improve at least a 5% 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 sub-problems the total number of objects by looking at the confusion matrices. For simplicity and fast computation, the used splitting criterion is  $k$ -means with  $k=2$ .<sup>3</sup> The results of some UCI data sets for *NMC* are shown graphically in fig. 4. One can see that the results of the sub-class 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. The mean rank of each ECOC design for each base classifier and for the whole set of UCI problems are numerically shown in table 4<sup>4</sup>. 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. Observing the ranks of each ECOC design for each base classifier and the global rank, one can see that the Sub-class approach attains the best position, and thus, performance, in all cases.

**Table 4.** Rank positions of the classification strategies for the UCI experiments.

	one-versus-one	one-versus-all	dense	sparse	DECOC	Sub-class ECOC
Discrete Adaboost	2.2	3.2	2.6	3.5	2.2	<b>1.3</b>
NMC	2.2	4.7	5.0	5.2	2.6	<b>1.1</b>
FLDA	1.6	3.8	3.1	3.8	2.1	<b>1.3</b>
Linear SVM	2.1	3.5	3.3	3.2	1.8	<b>1.0</b>
RBF SVM	2.3	4.2	2.6	4.3	2.6	<b>1.2</b>
Global rank	2.1	3.9	3.3	4.0	2.3	<b>1.2</b>

### 3.3 Traffic sign categorization

For this experiment, we use the video sequences obtained from the Mobile Mapping System [5] to test a real traffic sign categorization problem. We choose the speed data set since the low resolution of the image, the non-controlled conditions, and the high similarity among classes make the categorization a difficult

<sup>3</sup> It is important to save the history of splits to re-use the sub-groups 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.

<sup>4</sup> We realize that averaging over data sets has a very limited meaning as it entirely depends on the selected set of problems.

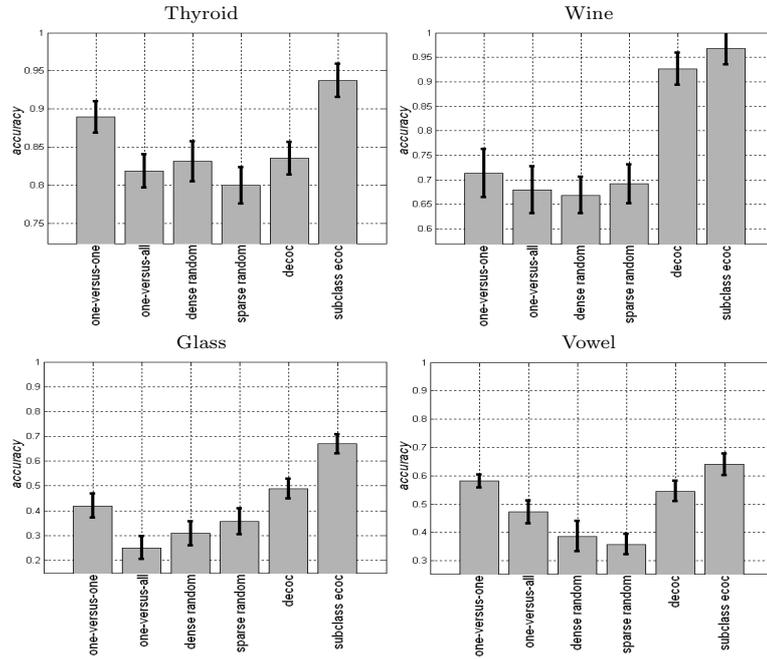


Fig. 4. UCI experiments for *NMC*.

task. In this system, the position and orientation of the different traffic signs are measured with fixed video cameras in a moving vehicle [5]. Fig. 5 shows several samples of the speed data set used for the experiments. The data set contains a total of 2500 samples divided into nine classes. Each sample is composed by 1200 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% of the total variance.



Fig. 5. Speed data set samples.

The performance and the estimated ranks using the different ECOC strategies for the different base classifiers are shown in table 5. One can see that in this particular problem, the sub-class 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 sub-classes. 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 Sub-class ECOC approach attains a better position in the global rank of table 5.

**Table 5.** Rank positions of the classification strategies for the Speed data set.

	one-versus-one	one-versus-all	dense	sparse	DECOC	Sub-class ECOC
D. 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

## 4 Conclusions

The Sub-class ECOC strategy presents a novel way to model complex multi-class classification problems. The method is based on embedding dichotomizers in a problem-dependent ECOC design by considering sub-sets 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 sub-class strategy. Sequential Forward Floating Search based on maximizing the Mutual Information is used to generate sub-groups of problems that are split until the desired performance is achieved. The experimental results over a set of UCI data sets and on a real multi-class traffic sign categorization problems for different base classifiers over the state-of-the-art ECOC configurations show the utility of the present methodology.

## References

1. OSU-SVM-TOOLBOX [<http://svm.sourceforge.net>].
2. 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.
3. T. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes", *JAIR*, vol. 2, pp. 263-286, 1995.
4. A. Asuncion and D.J. Newman, UCI Machine Learning Repository", University of California, Irvine, School of Information and Computer Sciences, 2007.
5. J. Casacuberta, J. Miranda, M. Pla, S. Sanchez, A.Serra and J.Talaya, "On the accuracy and performance of the geomobil system", *International Society for Photogrammetry and Remote Sensing*, 2004.
6. O. Pujol, P. Radeva, and J. Vitrià, "Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes", *Trans. on PAMI*, vol. 28, pp. 1001-1007, 2006.
7. O. Pujol, S. Escalera, and P. Radeva, "An Incremental Node Embedding Technique for Error Correcting Output Codes", *Pattern Recognition*, to appear.
8. E. Allwein, R. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers", *JMLR*, vol. 1, pp. 113-141, 2002.
9. S. Escalera, O. Pujol, and P. Radeva, "Loss-Weighted Decoding for Error-Correcting Output Codes", *CVCRD*, pp. 77-82, October 2007.
10. H. Daume and D. Marcu, "A Bayesian Model for Supervised Clustering with the Dirichlet Process Prior", *JMLR*, vol. 6, pp. 1551-1577, 2005.
11. M. Zhu and A. M. Martinez, "Subclass Discriminant Analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, issue 8, pp. 1274-1286, 2006.
12. P. Pudil, F. Ferri, J. Novovicova, and J. Kittler, "Floating Search Methods for Feature Selection with Nonmonotonic Criterion Functions", *Proc. Int. Conf. Pattern Recognition*, pp. 279-283, 1994.