Discriminant ECOC: A Heuristic Method for Application Dependent Design of Error Correcting Output Codes

Oriol Pujol, Petia Radeva, *Member*, *IEEE*, and Jordi Vitrià

Abstract—We present a heuristic method for learning error correcting output codes matrices based on a hierarchical partition of the class space that maximizes a discriminative criterion. To achieve this goal, the optimal codeword separation is sacrificed in favor of a maximum class discrimination in the partitions. The creation of the hierarchical partition set is performed using a binary tree. As a result, a compact matrix with high discrimination power is obtained. Our method is validated using the UCI database and applied to a real problem, the classification of traffic sign images.

Index Terms—Multiple classifiers, multiclass classification, visual object recognition.

1 Introduction

THE task of supervised machine learning can be seen as the problem of finding an unknown function $C(x)$ given the training set of example pairs $< \mathbf{x}_i, C(\mathbf{x}_i) >$. $C(x)$ is usually a set of discrete labels. For example, in face detection, $C(x)$ is a binary function $C(x) \in \{\text{face}, \text{nonface}\}$, in optical digit recognition $C(x) \in \{0, \ldots, 9\}$.

In order to address the binary classification task many techniques and algorithms have been proposed: decision trees, neural networks, large margin classification techniques, etc. Some of those methods can be easily extended to multiclass problems. However, some other powerful and popular classifiers, such as AdaBoost [5] and Support Vector machines [3], do not extend to multiclass easily. In those situations, the usual way to proceed is to reduce the complexity of the multiclass problem into multiple simpler binary classification problems.

There are many different approaches for reducing multiclass to binary classification problems. The simplest approach considers the comparison between each class against all the others. This produces N_c binary problems, where N_c is the number of classes. Other researchers suggested the comparison of all possible pairs of classes [6], resulting in an $N_c(N_c-1)/2$ set of binary problems. Dietterich and Bakiri [8] presented a general framework in which the classification is performed according to a set of binary error correcting output codes (ECOC). In this approach, the problem is transformed in n binary classification subproblems, where n is the error correcting output code length $n \in \{N_c, \ldots, \infty\}$. Then, the output of all classifiers must be combined—traditionally using Hamming distance. The approach of Dietterich and Bakiri was improved by Allwein et al. [7] by introducing an uncertainty value in the ECOC design and exploring alternatives for mixing the resulting outputs of the classifiers. In particular, they introduced loss-based decoding as a way of merging the classifiers. Recently, Passerini et al. [2] proposed a new decoding function that combines the margins through an estimate of the class conditional probabilities. ECOC strategies have been proven to be quite competitive with/better than other multiclass extensions of SVM and Adaboost [15], [16].

Although most of the improvements in error correcting output codes have been made in the decoding process, little attention has been paid to the design of the codes themselves. Crammer and Singer in [1] were the first to report improvements in the design of the codes. However, the results were rather pessimistic since they proved that the problem of finding the optimal discrete codes is computationally intractable since it is NP-complete.

It is our purpose in this paper to reopen the problem of designing the discrete ECOC by proposing a heuristic method that not only gives an efficient and effective method for ECOC design, but leads to compact codes of $N_c - 1$ bits (binary problems).

The method we propose renders each column of the output code matrix to the problem of finding the binary partition that divides the whole set of classes so that the discriminability between both sets is maximum. The criterion used for achieving this goal is based on the mutual information between the feature data and its class label. Since the problem is defined as a discrete optimization process, we propose using the floating search method as a suboptimal search procedure for finding the partition that maximizes the mutual information. The whole ECOC matrix is created with the aid of an intermediate step formulated as a binary tree. With this formulation, we ensure that we decompose the multiclass problem into $N_c - 1$ binary problems.

The paper is divided in the following sections: Section 2 provides a brief introduction to error correcting output codes, Section 3 describes the discriminant ECOC technique as well as the theory of the methods involved in its creation. Section 4 shows empirical results of the proposed method and Section 5 concludes the paper.

2 Error Correcting Output Codes

Error correcting output codes were born as a general framework for handling multiclass problems [8]. The basis of this framework is to create a codeword for each class (up to N_c codewords). Arranging the codewords as rows of a matrix, they define the "coding matrix" M, where $M \in \{-1, 1\}^{N_c \times n}$ and n is the code length.

From the point of view of learning, the matrix M is interpreted as a set of n binary learning problems, one for each column. Each column defines a partition of classes (coded by +1, -1 according to their class membership). As a result of the outputs of the n binary classifiers (dichotomies from now on) a code is obtained for each data point in the test set. This code is compared with the base codewords of each class defined in the matrix M and the data point is assigned to the class with the "closest" codeword.

A generalization of this process is provided in [7]. The main difference in terms of the coding matrix is that it is taken from a larger set $M \in \{-1, 0, 1\}^{N_c \times n}$. In this approach, some entries in the matrix M can be zero indicating that a particular class is not significative for a given dichotomy. In practical applications, this means that each dichotomy omits all examples for which $M = 0$. As we mentioned before, the codeword is formed by applying the different dichotomies to a given instance x and concatenating the results from each of the dichotomies. If we denote $f(x) = (f_1(x), \ldots, f_n(x))$ the vector of predictions for the sample x, the combination of the n outputs assigns one of the N_c labels. The simplest way of decoding a vector $f(x)$ is the *Hamming decoding*. This method looks for the minimum distance $d_H(M(r,.), f(x))$ between the prediction and the codewords:

$$\hat{y} = argmin_r \quad (d_H(M(r,.), f(x))),$$

$$d_H(M(r,.), f(x)) = \sum_{s=1}^{n} \left(\frac{1 - \text{sign}(M(r,s) f_s(x))}{2} \right),$$

where $\text{sign}(z)$ is $+1$ if $z > 0$, -1 if $z < 0$ and 0 otherwise. $M(r,.)$ designates the codeword r in the matrix and $\hat{y} \in \{1, \ldots, N_c\}$ is the predicted label.

Table 1 provides two examples of M matrices applied to a four class problem. C_i is the class label and h_i is the dichotomy learner. In the case of one-against-all classification, M is an $N_c \times N_c$ matrix

- *O. Pujol is with the Departament de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via 585, Barcelona, 08007, Spain. E-mail: oriol@maia.ub.es.*
- *P. Radeva and J. Vitrià are with the Departament de Ciències de la Computació and Computer Vision Center, Universitat Autònoma de Barcelona, Edifici O, Campus UAB, Bellaterra, 08193, Spain. E-mail: {petia, jordi}@cvc.uab.es.*

Manuscript received 27 May 2005; revised 1 Dec. 2005; accepted 12 Dec. 2005; published online 13 Apr. 2006.
Recommended for acceptance by B.S. Manjunath.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0270-0505.

TABLE 1
Example of the M Matrices for a 4-Class Problem
(a) 1-against-All matrix and (b) All-Pairs Matrix

	h1	h2	h3	h4
C1	+1	-1	-1	-1
C2	-1	+1	-1	-1
C3	-1	-1	+1	-1
C4	-1	-1	-1	+1

(a)

	h1	h2	h3	h4	h5	h6
C1	+1	+1	+1	0	0	0
C2	-1	0	0	+1	+1	0
C3	0	-1	0	-1	0	+1
C4	0	0	-1	0	-1	-1

(b)

in which all diagonal elements are set to +1 while the rest are set to -1. In the case of all-pairs classifiers, M is an $N_c \times N_c(N_c-1)/2$ matrix in which each column is set to zero except for a given pair. One of the pair elements is set to 1 and the other to -1.

Several other heuristics for creating ECOC matrices are proposed in [8] and [7] such as sparse random codes and dense random codes. In the *dense random codes*, each element in the code is chosen uniformly at random from the set $\{+1,-1\}$. Allwein et al. [7] suggested an optimal length of $10log_2(N_c)$ bits per code. The dense matrix is created by choosing the matrix that has the largest minimum Hamming decoding distance among each pair of codewords in the matrix—the matrix with trivial and complementary codes is discarded. The second random approach—*sparse random codes*—takes its values from the pool $\{+1,0,-1\}$. Each element of the coding matrix is 0 with probability $\frac{1}{2}$ and -1 or 1 with probability $\frac{1}{4}$ each. The length of the sparse codeword is set to $15log_2(N_c)$. Again, we choose the matrix with the largest minimum Hamming decoding distance considering that no trivial or complementary codes are present.

All these codification strategies are defined independently of the data set and satisfy two properties:

- **Row separation**. Each codeword should be well-separated in Hamming decoding distance from each of the other codewords.
- **Column separation**. Each column h_i should be uncorrelated with all the other columns $h_j, j \neq i$. This property is achieved if the Hamming decoding distance between a column and the rest—including their complementaries—is large.

Up to now, the priority when designing ECOC matrices was set in the codeword separation. Our approach trades part of the error correcting capabilities for classifier performance in each dichotomy.

3 Discriminant ECOC

Discriminant ECOC is born as an answer to three demands: First, a heuristic for the design of the ECOC matrix, second, the search for high-performance classification using the minimum number of classifiers, and, third, a tool to describe the classification domain in terms of class dependencies. We have seen in the previous section that one-against-all and dense random codes, on one hand, and all-pairs and sparse random codes on the other, are classic examples for binary and ternary valued ECOC designs, respectively. Our approach relaxes the fixed topologies of the matrices of the one-against-all and all-pairs strategies by allowing the classes to organize in maximally discriminant sets. Besides, we consider that this meaningful organization helps in keeping the number of classifiers low—as opposed to the high number of classifiers needed for the random strategies to show good performance. As a result of these specifications, we define the discriminant ECOC.

3.1 Design of the Discriminant ECOC

The goal of this work is to find a compact—in terms of codeword length—matrix M with high discriminative power. The general algorithm can be described as follows:

General procedure

- Create the *Column Code Binary Tree*—recursively, find the most discriminant binary partition of each parent node class set—$\{\wp_k^1, \wp_k^2\}$—using *floating search* with *fast quadratic mutual information* criterion.
- Assign to the column k of matrix M the code obtained by the partition $\{\wp_k^1, \wp_k^2\}$.

The first step is the creation of the *Column Code Binary Tree* (CCBT), where each node of the tree defines a partition of the classes. The partition at each node must satisfy the condition of being highly separable in terms of discrimination. This division is obtained as the result of the maximization of the quadratic mutual information between the data $\mathbf{x}$ and the labels created for such partition d. The algorithm used for the discrete maximization is the floating search method, which will be introduced in the next section.

Table 2 shows the basic algorithm for creating the code column binary tree (CCBT). In the algorithm, d is a discrete random variable, so that, given a binary partition $\{\wp_k^1, \wp_k^2\}$ of the set S_k, $\{\wp_k^1, \wp_k^2\} = BP(S_k)$, d is defined in the following terms,

$$d = d(\mathbf{x}, BP(S_k)) = \begin{cases} 1 & \text{if } \mathbf{x} \in C_i | C_i \in \wp_k^1 \\ -1 & \text{if } \mathbf{x} \in C_i | C_i \in \wp_k^2. \end{cases}$$

The tree must be seen as a means to finding the codewords. The second step is the process of filling the ECOC matrix. The final matrix M is composed by the codes obtained at each node—except for the leaves. Those codes are placed as columns in the coding matrix, $M(.,i)$. In order to create each column code, we use the relationship between a parent node and its children. Therefore, given a certain class C_r and the class set associated to node k: $\{\wp_k^1 \cup \wp_k^2\}$ (where $\wp_k^1$ and $\wp_k^2$ are the sets of classes for each one of the children of the node k, respectively), matrix M is filled as follows:

$$M(r,i) = \begin{cases} 0 & \text{if } C_r \notin \wp_i \\ +1 & \text{if } C_r \in \wp_i^1 \\ -1 & \text{if } C_r \in \wp_i^2. \end{cases}$$

Note that the number of columns—n—coincides with the number of internal nodes. It is easy to see that, in any binary tree, the number of internal nodes is $N_c - 1$ given that the number of leaves is N_c. Therefore, by means of the CCBT, we can assure that the codeword will have length $N_c - 1$.

Fig. 1 shows an example of a CCBT for eight classes. On the right side of the figure, we show the resulting discriminant ECOC matrix. The white squares are +1, black squares are -1, and gray squares have 0 value. Observe, for instance, that column $N5$ corresponds to the partition $\wp_5^1 = \{c5, c6\}$ and $\wp_5^2 = \{c2\}$. On the other hand, if we look at the rows of the matrix, the codeword associated to class 6 ($c6$) is $\{+1,0,-1,0,-1,0,+1\}$.

From a more general point of view, the creation of the ECOC matrix is only one of the parts involved in the multiclass classification technique. The other two remaining parts to be defined are the dichotomy learning technique and the decoding strategy. In this paper, we have chosen decision trees and AdaBoost [5] as base classifiers for each dichotomy. The chosen decoding metric is the Euclidean distance to the codewords.[1] Moreover, it can be shown that Euclidean and Hamming decoding distances have the same performance for standard ECOC strategies.

1. The main drawback of Hamming decoding distance is the ambiguity that appears due to the fact that we are using the zero symbol. Suppose that we have two codewords that are exactly equal except for two bits. Codeword A contains the following code $\{X,X,X,X,X,X,X,X,0,0\}$ and codeword B is coded as $\{X,X,X,X,X,X,X,X,1,-1\}$, where the X are whatever values we like but the same for both codewords. Assume now that as a result of our test, we obtain the codeword $TEST = \{X,X,X,X,X,X,X,X,1,1\}$. The Hamming decoding values of $TEST$ to A and $TEST$ to B are exactly the same. This creates an ambiguous choice that is usually solved by random tie-breaking.

TABLE 2
CCBT Algorithm

[Initialization:]

Create the trivial partition $\{\wp_0^1, \wp_0^2\}$ of the set of classes $\{C_i\}$:

$\{\wp_0^1, \wp_0^2\} = \{\{\emptyset\}, \{C_1, C_2, \ldots, C_{N_c}\}\}$

$L_0 = \{\wp_0^2\}$

$k = 1$

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 $BP(S_k)$:

$\{\bar{\wp}_k^1, \bar{\wp}_k^2\} = \underset{BP(S_k)}{argmax}(I(\mathbf{x}, d(BP(S_k))))$

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]

Step 3. $L_k = \{L'_k \cup \bar{\wp}_k^i\}$ if $|\bar{\wp}_k^i| > 1 \quad \forall i \in \{1, 2\}$

Step 4. If $|L_k| \neq 0$

$k = k + 1$ go to **Step 1**

[a]Use *floating search* as the maximization procedure and *fast quadratic mutual information* to estimate I

Recalling the algorithm described in Table 2, a maximization process is needed to obtain the division of the classes in two sets. Although looking for the best partition set requires of an exhaustive search among all possible partitions, due to the impracticability of this endeavor a suboptimal strategy must be used. The strategy chosen is the *floating search method*. The following subsection details this method that allows the problem to be computationally feasible.

3.2 Floating Search Methods

The *Floating search method* [9] was born as a suboptimal sequential search method for alleviating the prohibitive computation cost of exhaustive search methods in feature selection. Furthermore, these methods allowed the search criterion to be nonmonotonic, thus solving the main constraint of many sequential methods. Floating search methods can be described as a dynamically changing number of forward steps and backward steps as long as the resulting subsets are better than the previously evaluated ones at that level. In this sense, this method avoids nesting effects that are typical of sequential forward and backward selection while being equally step-optimal since the best (worst) item is always added (discarded) to (from) the set. The algorithm presented in Table 3 describes the top-down approach which is called Sequential Forward Floating Search (SFFS) algorithm. This one 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. In our case, Y is the set of classes to be partitioned. The criterion used for dividing the class set is related to the discriminability. In particular, we use mutual information to that

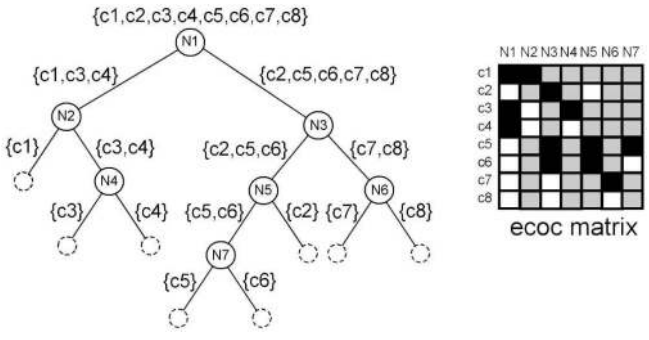

Fig. 1. Example of conversion from the binary tree to the ECOC matrix.

TABLE 3
SFFS Algorithm

Input:

$Y = \{y_j | j = 1..D\}$//Available items//

Output:

$X_k = \{x_j | j = 1..|Y| (or D), x_j \in Y\}$

Initialization:

$X_0 = \{\emptyset\}; \quad k = 0$

Termination:

Stop when the criterion does not increase $J(X_k) \approx J(X_{k-1})$

Step 1 **(Inclusion)**

$x^+ = \underset{x \in Y - X_k}{argmax}\ J(X_k \cup x)$

$X_{k+1} = X_k \cup x^+, \quad k = k + 1$

Step 2 **(Conditional exclusion)**

$x^- = \underset{x \in X_k}{argmax}\ J(X_k - x)$

if $J(X_k - x^-) > J(X_{k-1})$ then

$X_{k+1} = X_k - x^-, \quad k = k + 1$

go to **Step 2**

else

go to **Step 1**

effect. Our goal is to maximize the mutual information between the data in the sets and the class labels created for each subset.

3.3 Fast Quadratic Mutual Information

Mutual information (MI) is a well-known criterion to compute the amount of information that one random variable tells about another one. In classification theory, this measure has been shown to be optimal in terms of class separation [12], [11], allowing to take into account high-order statistics. MI also bounds the optimal Bayes error rate. However, mutual information is not widely used due to the difficulties derived from its computation.

Although evaluating MI in low-dimensional spaces—small number of features—can be feasible through histograms, it cannot be easily accomplished in high-dimensional ones due to sparsity of data. However, Principe et al. [11] presented a feasible method for computing entropy estimators using Renyi's formulation when coupled with Parzen window density estimation. Based on this method, they heuristically obtained a measure for mutual information. This work has been recently modified and extended by Torkkola [12] by relating mutual information to divergence measures. Using this extension, the authors provide the base for computing "quadratic mutual information" in a simple and fast way.

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

$$I(\mathbf{x},\mathbf{y}) = \int\int p(x,y) log\left(\frac{p(x,y)}{p(x)p(y)}\right) dxdy. \quad (1)$$

Observe that mutual information is zero if $p(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$. It is important to note that (1) can be seen as a Kullback-Leibler divergence,

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

where $f(y)$ is replaced with $p(x,y)$ and $g(y)$ with $p(x)p(y)$.

Alternatively, Kapur and Kesavan argued in [10] 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 proven in [12] that maximizing $K(f,g)$ is equivalent to maximizing $D(f,g)$. Therefore, we can define the quadratic mutual information as,

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

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

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

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

In our problem, the mutual information is computed between the random variable of the features $\mathbf{x}$ and the discrete random variable associated to the class labels created for a given partition ($\mathbf{d}$). Let us define the notation for the practical implementation of I_Q: Assume that we have N samples in the whole data set, J_p are the samples of each class p, N_c stands for the number of classes, x_l stands for the lth feature vector of the data set, and x_{pk} is the kth feature vector of the set in class p. Then, $p(\mathbf{d})$ and $p(\mathbf{x}|\mathbf{d})$ can be written as:

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

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

Expanding (2) and using a Parzen estimate with a symmetric kernel with width σ, we obtain the following equations,

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

where

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

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

$$V_{BTW} = \sum\int p(\mathbf{x},\mathbf{d}) p(\mathbf{x}) p(\mathbf{d}) dx = \frac{1}{N^2}\sum_{p=1}^{N_C}\frac{J_p}{N}\sum_{l=1}^{N}\sum_{k=1}^{J_p} N(x_l-x_{pk},2\sigma^2 I). \quad (3)$$

In practical applications, σ is usually set to the half of the maximum distance between samples as proposed by Torkkola in [12].

4 Experimental Results

In this section, we describe and discuss the experiments we have performed with data from two different environments. First, we validate the approach using data from the UCI repository. Afterward, we apply this approach to a real problem: traffic sign recognition.

4.1 Validation on UCI Database

To validate our approach, we begin with an analysis using the standard UCI database [17]. This database is a well-known database for evaluation and comparison of classifiers. We have chosen two very popular binary classifiers for these experiments in order to empirically demonstrate that the advantages of the method are not related to the base classifier (Adaboost, decision trees, etc.). The first one, AdaBoost [5], has 40 weak learners per strong dichotomy (h_j). The weak learner is a *decision stump* [13]. The second one is a decision tree [4] using a purity split criterion (Gini value) with an early stopping value set at 3. We have selected from the UCI database the following data sets:

1. abalone,
2. dermatology,
3. e. coli,
4. glass,
5. yeast,
6. iris,
7. vowel-general,
8. balance-scale, and
9. wine.

The properties of the data sets are described in Table 4. The experiments have been performed using a 10-fold cross-validation strategy. We compare our approach with all-pairs, one-against-all, dense random codes, and sparse random codes. In both random approaches, we have examined 5,000 different matrices. Table 5 and Table 6 show the results for Adaboost and decision trees, respectively. The tables display the resulting mean error rate and the confidence interval at 95 percent; we have tested for statistical significance using a two-tailed t-test [14]. The tables must be read in the following way: There is an * marker on the right side of the result that achieves the absolute lowest mean error rate. In bold face, the results with mean error rate not statistically significant from the highest performance method are highlighted. The table also shows the mean rank value according to the error rate. An alternate rank taking into account the confidence interval is displayed in brackets. The alternate rank considers that all classifiers with a mean error rate not statistically significant from the top ranked method are rated as first choice.

TABLE 4
Description of the Data Sets Used in the Experiments

Problem	#Examples	#Attributes	#Classes	Problem	#Examples	#Attributes	#Classes
iris	150	4	3	glass	214	9	7
wine	178	13	3	ecoli	336	8	8
balance-scale	625	4	3	yeast	1484	8	10
dermatology	366	34	6	vowel	998	10	11
abalone	4177	8	28				

TABLE 5
Error Rate and Confidence Interval at 95 Percent for Several UCI Data Sets Using Adaboost

	DECOC	1 vs 1	1 vs All	Dense	Sparse
((a)	$76.32 \pm 0.37\%$	$\mathbf{74.20 \pm 0.41\%}$*	$97.85 \pm 0.45\%$	$75.21 \pm 0.44\%$	$\mathbf{74.31 \pm 0.41\%}$
(b)	$\mathbf{5.66 \pm 0.76\%}$*	$\mathbf{6.63 \pm 0.81\%}$	$11.43 \pm 1.01\%$	$7.79 \pm 0.80\%$	$8.36 \pm 0.96\%$
(c)	$\mathbf{16.35 \pm 1.18\%}$*	$18.48 \pm 1.2\%$	$24.04 \pm 1.47\%$	$\mathbf{16.85 \pm 1.25\%}$	$\mathbf{16.42 \pm 1.28\%}$
(d)	$\mathbf{27.02 \pm 2.01\%}$*	$\mathbf{27.15 \pm 1.65\%}$	$34.11 \pm 2.16\%$	$31.51 \pm 1.79\%$	$30.19 \pm 2.28\%$
(e)	$\mathbf{43.52 \pm 0.72\%}$*	$46.17 \pm 0.77\%$	$48.28 \pm 0.76\%$	$47.88 \pm 0.72\%$	$49.03 \pm 0.74\%$
(f)	$\mathbf{5.47 \pm 0.94\%}$	$\mathbf{5.23 \pm 0.99\%}$*	$\mathbf{5.64 \pm 1.14\%}$	$\mathbf{5.26 \pm 1.13\%}$	$\mathbf{6.15 \pm 1.17\%}$
(g)	$19.51 \pm 0.73\%$	$\mathbf{17.52 \pm 0.83\%}$*	$51.54 \pm 1.17\%$	$36.14 \pm 0.92\%$	$31.55 \pm 0.88\%$
(h)	$\mathbf{8.61 \pm 0.68\%}$	$8.87 \pm 0.81\%$	$\mathbf{8.04 \pm 0.62\%}$	$\mathbf{7.68 \pm 0.29\%}$*	$8.94 \pm 0.79\%$
(i)	$\mathbf{5.56 \pm 1.08\%}$	$6.16 \pm 1.12\%$	$7.33 \pm 1.04\%$	$\mathbf{3.48 \pm 1.01\%}$*	$\mathbf{3.69 \pm 1.09\%}$
Rank	**2.11(1.33)**	2.33 (1.44)	4.44 (3.22)	2.66 (2.00)	3.44 (2.22)

TABLE 6
Error Rate and Confidence Interval at 95 Percent for Several UCI Data Sets Using Decision Trees

	DECOC	1 vs 1	1 vs All	Dense	Sparse
(a)	84.72 ± 0.35	$\mathbf{82.12 \pm 0.42}$*	97.12 ± 0.39	84.88 ± 0.41	84.16 ± 0.43
(b)	12.61 ± 0.76	12.94 ± 0.85	$\mathbf{12.20 \pm 1.01}$	$\mathbf{11.01 \pm 0.78}$	$\mathbf{10.59 \pm 0.89}$*
(c)	$\mathbf{25.12 \pm 1.11}$	$\mathbf{24.49 \pm 1.03}$*	29.10 ± 1.58	37.70 ± 1.56	31.87 ± 1.42
(d)	$\mathbf{44.04 \pm 1.23}$*	$\mathbf{44.63 \pm 1.41}$	49.23 ± 1.58	$\mathbf{45.37 \pm 1.85}$	50.58 ± 1.75
(e)	$\mathbf{64.30 \pm 0.91}$	$\mathbf{62.59 \pm 0.88}$*	68.18 ± 0.91	68.64 ± 0.81	67.83 ± 0.87
(f)	$\mathbf{8.52 \pm 1.01}$	$\mathbf{7.57 \pm 1.03}$	$\mathbf{8.60 \pm 1.49}$	$\mathbf{7.56 \pm 0.98}$*	$\mathbf{9.28 \pm 1.23}$
(g)	29.46 ± 0.92	$\mathbf{18.61 \pm 0.88}$*	67.08 ± 1.14	64.79 ± 0.75	61.39 ± 0.87
(h)	$\mathbf{21.76 \pm 0.86}$	23.86 ± 0.75	24.96 ± 0.93	23.25 ± 0.56	$\mathbf{21.04 \pm 0.80}$*
(i)	$\mathbf{6.38 \pm 1.12}$	9.43 ± 1.07	$\mathbf{7.14 \pm 1.04}$	$\mathbf{7.67 \pm 1.01}$	$\mathbf{5.84 \pm 1.01}$*
Rank	**2.33(1.44)**	2.44(1.55)	4.11(2.77)	3.33(2.33)	2.77(1.88)

Examining the tables, we can see that the DECOC technique not only provides more compact codes in terms of codeword length—hence, it reduces the training and test time—but it also achieves a better overall result and is ranked as the preferred method in both experiments. This merit is further increased if we take into account that the only method that can compete fairly—in terms of complexity—is the one-against-all strategy. Precisely, looking at the figures of the tables, one can observe that one-against-all performs very poorly. We must take into account that this technique easily falls in imbalanced problems, e.g., in the Abalone or Vowel data sets. Although the experiments show a very good behavior of our method, other techniques such as all-pairs and random codes follow very closely and must be taken into account for extending binary classifiers to multiclass ones.

4.2 Traffic Sign Recognition

The proposed approach was used in an online traffic sign detection and recognition project for guided navigation. In particular, we are concerned with the traffic sign recognition part. In this problem, we have a set of 32 different signs that have to be distinguished. An example of each class is illustrated in Fig. 2a. We used the five different approaches to compare the performance in this problem. The binary base classifier is AdaBoost with 40 decision stumps as weak learners. The training set was extracted from eight car drive

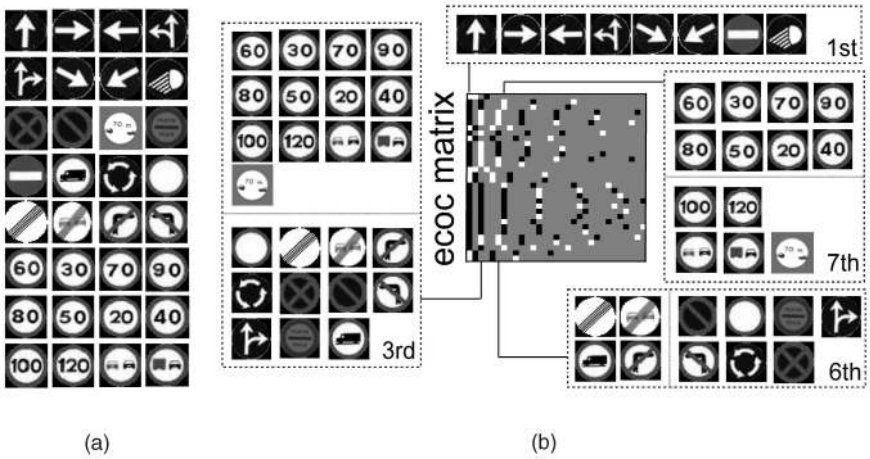

Fig. 2. (a) The 32 different signal classes to recognize. (b) Discriminant ECOC matrix created for the signal recognition system and partitions at the first, third, sixth, and seventh columns.

records at different locations, highways, and local roads. The total number of examples sums to 2,217. This problem has an additional difficulty since the number of samples for each class is very different. This means that we are faced with an imbalanced class problem with classes clearly underrepresented. Each extracted sign image measures 35×35 pixels. Each pixel is considered as an object feature. Thus, the feature vector dimensionality is $1,225$. We have used two different car drive records (data set1 and data set2, corresponding to a highway and a local road, respectively) as test sets. The total number of traffic signs in the test records sums to 600. The results obtained in our experiments are summarized in Table 7. We can observe that the behavior of the five methods follows the guidelines obtained in the validation of the method using the UCI datasets. Note the successful performance of the DECOC technique in this real problem. This success is reinforced by the fact that our approach uses only 31 classifiers instead of the 496 classifiers required in the all-pairs approach; thus, increasing the computational efficiency of the whole process, training, and test. Fig. 2b shows the discriminant ECOC matrix for the signal recognition application along with some partitions of the classes. The partitions shown correspond to the first, third, sixth, and seventh column of the matrix. Note that, in the first partition, only the smaller group is displayed—the other group is composed by the rest of the traffic signs.

5 Conclusion

We introduced a new algorithm, discriminant ECOC, for designing compact error correcting output codes. The result is a multiclass classifier that runs faster—since it uses fewer classifiers—and requires less training time, while maintaining—and improving in some cases—the performance of the rest of ECOC approaches. This methodology is also the first one to deal successfully with the problem of the design of application dependent discrete ECOC matrices. Discriminant ECOC algorithm has been applied successfully to two problems: First, the UCI database for validation purposes and, second, to a real computer vision application, traffic sign recognition. From the different experiments, we observe that the building process of the ECOC matrix is of great importance. We can conclude that the discriminant ECOC design is a very promising alternative to other ECOC methods, frequently outperforming most of them. Our current line of research is centered on the enrichment of the discriminant ECOC matrix by embedding different tree structures to form a possible ECOC forest.

TABLE 7
Error Rates for Traffic Sign Recognition

Test sequence	DECOC	1 vs 1	1 vs All	Dense	Sparse
dataset1	**9.38%**	13.14%	27.23%	9.85%	19.24%
dataset2	**12.68%**	15.85%	29.17%	24.10%	24.52%

Acknowledgments

This work has been supported by FIS: PI031488, FIS network: G03/185 of MEC and MCYT grant TIC2003-00654.

References

[1] K. Crammer and Y. Singer, "On the Learnability and Design of Output Codes for Multiclass Problems," *Machine Learning*, vol. 47, no. 2-3, pp. 201-233, 2002.
[2] A. Passerini, M. Pontil, and P. Frasconi, "New Results on Error Correcting Codes of Kernel Machines," *IEEE Trans. Neural Networks*, vol. 15, no. 1, pp. 45-54, 2004.
[3] V.N. Vapnik, *The Nature of Statistical Learning Theory*. Springer 1995.
[4] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986.
[5] Y. Freund and R.E. Shapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, 1997.
[6] T. Hastie and R. Tibshirani, "Classification by Pairwise Coupling," *Annals of Statistics*, vol. 26, no. 2, pp. 451-471, 1998.
[7] E.L Allwein, R.E Shapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *J. Machine Learning Research*, vol. 1, pp. 113-141, 2000.
[8] T.G. Dietterich and G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," *J. Artificial Intelligence Research*, vol. 2, pp. 263-286, 1995.
[9] P. Pudil, F. Ferri, J. Novovicová, and J. Kittler, "Floating Search Methods for Feature Selection with Nonmonotonic Criterion Functions," *Proc. Int'l Conf. Pattern Recognition*, pp. 279-283, 1994.
[10] J.N. Kapur and H.K. Kesavan, *Entropy Optimization Principles with Applications*. London: Academic Press, 1992.
[11] J. Principe, D. Xu, and J. Fisher III, "Information Theoretic Learning," *Unsupervised Adaptive Filtering*, Wiley, 2000.
[12] K. Torkkola, "Feature Extraction by Non-Parametric Mutual Information Maximization," *J. Machine Learning Research*, vol. 3, pp. 1415-1438, 2003.
[13] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *Annals of Statistics*, vol. 28, no. 3, pp. 337-374, 2000.
[14] L. Kuncheva, *Combining Pattern Classifiers. Methods and Algorithms*. Wiley, 2004.
[15] R.E. Schapire, "Using Output Codes to Boost Multiclass Learning Problems," *Machine Learning: Proc. 14th Int'l Conf.*, pp. 313-321, 1997.
[16] C. Hsu and C. Lin, "A Comparison of Methods for Multi-Class Support Vector Machines," *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 415-425, Mar. 2002.
[17] P.M. Murphy and D.W. Aha, *UCI Repository of Machine Learning Databases*, Dept. of Information and Computer Science, Univ. of California, Irvine, 1994.