Mutual Information Measures for Subclass Error-Correcting Output Codes Classification

Nikolaos Arvanitopoulos, Dimitrios Bouzas, and Anastasios Tefas

Aristotle University of Thessaloniki, Department of Informatics
Artificial Intelligence & Information Analysis Laboratory
{niarvani,dmpouzas}@csd.auth.gr, tefas@aiia.csd.auth.gr

Abstract. *Error-Correcting Output Codes (ECOCs)* reveal a common way to model multi-class classification problems. According to this state of the art technique, a multi-class problem is decomposed into several binary ones. Additionally, on the ECOC framework we can apply the *subclasses technique (sub-ECOC)*, where by splitting the initial classes of the problem we aim to the creation of larger but easier to solve ECOC configurations. The multi-class problem's decomposition is achieved via a searching procedure known as *sequential forward floating search (SFFS)*. The SFFS algorithm in each step searches for the optimum binary separation of the classes that compose the multi-class problem. The separation decision is based on the maximization or minimization of a criterion function. The standard criterion used is the maximization of the *mutual information (MI)* between the bi-partitions created in each step of the SFFS. The materialization of the MI measure is achieved by a method called *fast quadratic Mutual Information (FQMI)*. Although FQMI is quite accurate in modelling the MI, its computation is of high algorithmic complexity, which as a consequence makes the ECOC and sub-ECOC techniques applicable only on small datasets. In this paper we present some alternative separation criteria of reduced computational complexity that can be used in the SFFS algorithm. Furthermore, we compare the performance of these criteria over several multi-class classification problems.

Keywords: Multi-class classification, Subclasses, Error-Correcting Output Codes, Support Vector Machines, Sequential Forward Floating Search, Mutual Information.

1 Introduction

In the literature one can find various binary classification techniques. However, in the real world the problems to be addressed are usually multi-class. In dealing with multi-class problems we must use the binary techniques as a leverage. This can be achieved by defining a method that decomposes the multi-class problem into several binary ones, and combines their solutions to solve the initial multi-class problem [1]. In this context, the Error-Correcting Output Codes (ECOCs)

S. Konstantopoulos et al. (Eds.): SETN 2010, LNAI 6040, pp. 19–28, 2010.

emerged. Based on the error correcting principles [2] and on its ability to correct the bias and variance errors of the base classifiers [3], this state of the art technique has been proved valuable in solving multi-class classification problems over a number of fields and applications.

As proposed by Escalera et al. [4], on the ECOC framework we can apply the subclass technique. According to this technique, we use a guided problem dependent procedure to group the classes and split them into subsets with respect to the improvement we obtain in the training performance. Both the ECOC and sub-ECOC techniques can be applied independently to different types of classifiers. In our work we applied both of those techniques on *Linear* and *RBF (Radial Basis Function) SVM (Support Vector Machine)* classifiers with various configurations. SVMs are very powerful classifiers capable of materializing optimum *classification surfaces* that give improved results in the *test domain*.

As mentioned earlier, the ECOC as well as the sub-ECOC techniques use the SFFS algorithm in order to decompose a multi-class problem into smaller binary ones. The problem's decomposition is based on a criterion function that maximizes or minimizes a certain quantity acording to the nature of the criterion used. The common way is to maximize the *MI (mutual information)* in both the bi-partitions created by SFFS. As proposed by Torkkola [5], we can model the MI in the bi-partitions through the *FQMI (Fast Quadratic Mutual Information)* method. However, although the FQMI procedure is quite accurate in modelling the MI of a set of classes, it turns out to be computational costly. In this paper we propose some novel MI measures of reduced computational complexity, where in certain classification problems yield better performance results than the FQMI. Furthermore, we compare these MI measures over a number of multi-class classification problems in the *UCI machine learning repository* [6].

1.1 Error Correcting Output Codes (ECOC)

Error Correcting Output Codes is a general framework to solve multi-class problems by decomposing them into several binary ones. This technique consists of two separate steps: a) the *encoding* and b) the *decoding* step [7].

a) In the encoding step, given a set of N classes, we assign a unique binary string called *codeword*[1] to each class. The length n of each codeword represents the number of *bi-partitions* (groups of classes) that are formed and, consequently, the number of binary problems to be trained. Each bit of the codeword represents the response of the corresponding binary classifier and it is coded by +1 or -1, according to its class membership. The next step is to arrange all these codewords as rows of a matrix obtaining the so-called *coding matrix* $\mathbf{M}$, where $\mathbf{M} \in \{-1, +1\}^{N \times n}$. Each column of this matrix defines a partition of classes, while each row defines the membership of the corresponding class in the specific binary problem.

[1] 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.

An extension of this standard ECOC approach was proposed by Allwein et al. [1] by adding a third symbol in the coding process. The new coding matrix $\mathbf{M}$ is now $\mathbf{M} \in \{-1, 0, +1\}^{N \times n}$. In this approach, the zero symbol means that a certain class is not considered by a specific binary classifier. As a result, this symbol increases the number of bi-partitions to be created in the ternary ECOC framework.

b) The decoding step of the ECOC approach consists of applying the n different binary classifiers to each data sample in the test set, in order to obtain a code for this sample. This code is then compared to all the codewords of the classes defined in the coding matrix $\mathbf{M}$ (each row in $\mathbf{M}$ defines a codeword) and the sample is assigned to the class with the closest codeword. The most frequently used decoding methods are the *Hamming* and the *Euclidean* decoding distances.

1.2 Sub-ECOC

Escalera et al. [4] proposed that from an initial set of classes $\mathcal{C}$ of a given multi-class problem, we can define a new set of classes $\mathcal{C}'$, where the cardinality of $\mathcal{C}'$ is greater than that of $\mathcal{C}$, that is $|\mathcal{C}'| > |\mathcal{C}|$. The new set of binary problems that will be created will improve the created classifiers' training performance. Additionally to the ECOC framework Pujol [8] proposed that we can use a ternary problem dependent design of ECOC, called *discriminant ECOC (DECOC)* where, given a number of N classes, we can achieve a high classification performance by training only $N - 1$ binary classifiers. The combination of the above mentioned methods results in a new classification procedure called sub-ECOC. The procedure is based on the creation of discriminant tree structures which depend on the problem domain.

These binary trees are built by choosing the problem partitioning that maximizes the MI between the samples and their respective class labels. The structure as a whole describes the decomposition of the initial multi-class problem into an assembly of smaller binary sub-problems. Each node of the tree represents a pair that consists of a specific binary sub-problem with its respective classifier. The construction of the tree's nodes is achieved through an evaluation procedure described in Escalera et al. [4]. According to this procedure, we can split the bi-partitions that consist the current sub-problem examined. Splitting can be achieved using K-means or some other clustering method. After splitting we form two new problems that can be examined separately. On each one of the new problems created, we repeat the SFFS procedure independently in order to form two new separate sub-problem domains that are easier to solve. Next, we evaluate the two new problem configurations against three user defined thresholds $\{\theta_p, \theta_s, \theta_i\}$ described below. If the thresholds are satisfied, the new created pair of sub-problems is accepted along with their new created binary classifiers, otherwise they are rejected and we keep the initial configuration with its respective binary classifier.

– θ_p: Performance of created classifier for newly created problem (after splitting).

- θ_s: Minimum cluster's size.
- θ_i: Performance's improvement of current classifier for newly created problem against previous classifier (before splitting).

1.3 Loss Weighted Decoding Algorithm

In the decoding process of the sub-ECOC approach we use the *Loss Weighted Decoding* algorithm [7]. As already mentioned, the 0 symbol in the decoding matrix allows to increase the number of binary problems created and as a result the number of different binary classifiers to be trained. Standard decoding techniques, such as the Euclidean or the Hamming distance do not consider this third symbol and often produce non-robust results. So, in order to solve the problems produced by the standard decoding algorithms, the loss weighted decoding was proposed.

The main objective is to define a *weighting matrix* $\mathbf{M}_W$ that weights a *loss function* to adjust the decision of the classifiers. In order to obtain the matrix $\mathbf{M}_W$, a *hypothesis matrix* $\mathbf{H}$ is constructed first. The elements $H(i, j)$ of this matrix are continuous values that correspond to the accuracy of the binary classifier h_j classifying the samples of class i. The matrix $\mathbf{H}$ has zero values in the positions which correspond to unconsidered classes, since these positions do not contain any representative information. The next step is the normalization of the rows of matrix $\mathbf{H}$. This is done, so that the matrix $\mathbf{M}_W$ can be considered as a discrete probability density function. This is very important, since we assume that the probability of considering each class for the final classification is the same. Finally, we decode by computing the weighted sum of our coding matrix $\mathbf{M}$ and our binary classifier with the weighting matrix $\mathbf{M}_W$ and assign our test sample to the class that attains the minimum *decoding value*.

1.4 Sequential Forward Floating Search

The *Floating search methods* are a family of suboptimal sequential search methods that were developed as an alternative counterpart to the more computational costly exhaustive search methods. These methods allow the search criterion to be *non-monotonic*. They are also able to counteract the nesting effect by considering conditional inclusion and exclusion of features controlled by the value of the criterion itself. In our approach we use a variation of the *Sequential Forward Floating Search (SFFS)* [9] algorithm. We modified the algorithm so that it can handle criterion functions evaluated using subsets of classes. We apply a number of backward steps after each forward step, as long as the resulting subsets are better than the previously evaluated ones at that level. Consequently, there are no backward steps at all if the performance cannot be improved. Thus, backtracking in this algorithm is controlled dynamically and, as a consequence, no parameter setting is needed.

The SFFS method is described in algorithm 1.

Algorithm 1. SFFS for Classes

Input:
$Y = \{y_j | j = 1, \ldots, N_c\}$ // available classes
Output: // disjoint subsets with maximum MI between the features and their class labels
$X_k = \{x_j | j = 1, \ldots, k,\ x_j \in Y\},\ k = 0, 1, \ldots, N_c$
$X'_{k'} = \{x_j | j = 1, \ldots, k',\ x_j \in Y\},\ k' = 0, 1, \ldots, N_c$
Initialization:
$X_0 := \emptyset, X'_{N_c} := Y; \quad k := 0, k' := N_c$ // k and k' denote the number of classes in each subset
Termination:
Stop when $k = N_c$ and $k' = 0$
Step 1 (*Inclusion*)
$x^+ := \arg \max_{x \in Y - X_k} J(X_k + x, X'_{k'} - x)$ { the most significant class *with respect to the group* $\{X_k, X'_{k'}\}$
$X_{k+1} := X_k + x^+;\ X'_{k'-1} := X'_{k'} - x^+;\ k := k + 1, k' := k' - 1$
Step 2 (*Conditional exclusion*)
$x^- := \arg \max_{x \in X_k} J(X_k - x, X'_{k'} + x)$ { the least significant class *with respect to the group* $\{X_k, X'_{k'}\}$
if $J(X_k - x^-, X'_{k'} + x^-) > J(X_{k-1}, X'_{k'+1})$ **then**
    $X_{k-1} := X_k - x^-;\ X'_{k'+1} := X'_{k'} + x^-;\ k := k - 1, k' := k' + 1$
    go to **Step 2**
else
    go to **Step 1**
end if

1.5 Fast Quadratic Mutual Information (FQMI)

Consider two random vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ and let $p(\mathbf{x}_1)$ and $p(\mathbf{x}_2)$ be their probability density functions respectively. Then the MI of $\mathbf{x}_1$ and $\mathbf{x}_2$ can be regarded as a measure of the dependence between them and is defined as follows:

$$\mathcal{I}(\mathbf{x}_1, \mathbf{x}_2) = \int \int p(\mathbf{x}_1, \mathbf{x}_2) \log \frac{p(\mathbf{x}_1, \mathbf{x}_2)}{p(\mathbf{x}_1)p(\mathbf{x}_2)} d\mathbf{x}_1 d\mathbf{x}_2 \tag{1}$$

Note that when the random vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ are stochastically independent, it holds that $p(\mathbf{x}_1, \mathbf{x}_2) = p(\mathbf{x}_1)p(\mathbf{x}_2)$.

It is of great importance to mention that (1) can be interpreted as a Kullback-Leibler divergence, defined as follows:

$$\mathcal{K}(f_1, f_2) = \int f_1(\mathbf{x}) \log \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} dx \tag{2}$$

where $f_1(\mathbf{x}) = p(\mathbf{x}_1, \mathbf{x}_2)$ and $f_2(\mathbf{x}) = p(\mathbf{x}_1)p(\mathbf{x}_2)$.

According to Kapur and Kesavan [10], if we seek to find the distribution that maximizes or alternatively minimizes the divergence, several axioms could be relaxed and it can be proven that $\mathcal{K}(f_1, f_2)$ is analogically related to $D(f_1, f_2) = \int (f_1(\mathbf{x}) - f_2(\mathbf{x}))^2 dx$. Consequently, maximization of $\mathcal{K}(f_1, f_2)$ leads to maximization of $D(f_1, f_2)$ and vice versa. Considering the above we can define the *quadratic mutual information* as follows

$$\mathcal{I}_Q(\mathbf{x}_1, \mathbf{x}_2) = \int \int (p(\mathbf{x}_1, \mathbf{x}_2) - p(\mathbf{x}_1)p(\mathbf{x}_2))^2 d\mathbf{x}_1 d\mathbf{x}_2 \tag{3}$$

Using *Parzen window estimators* we can estimate the probability density functions in (3) and combining with Gaussian kernels the following property is applicable: Let $\mathcal{N}(\mathbf{x}, \mathbf{\Sigma})$ be a n-dimensional Gaussian function; it can be shown that

$$\int \mathcal{N}(\mathbf{x} - \mathbf{a}_1, \mathbf{\Sigma}_1)\mathcal{N}(\mathbf{x} - \mathbf{a}_2, \mathbf{\Sigma}_2)d\mathbf{x} = \mathcal{N}(\mathbf{a}_1 - \mathbf{a}_2, \mathbf{\Sigma}_1 - \mathbf{\Sigma}_2) \tag{4}$$

and by the use of this property we avoid one integration.

In our case, we calculate the amount of mutual information between the random vector $\mathbf{x}$ of the features and the discrete random variable associated to the class labels created for a given partition (y). The practical implementation of this computation is defined as follows: Let N be the number of pattern samples in the entire data set, J_i the number of samples of class i, let N_c be the number of classes in the entire data set, let $\mathbf{x}_i$ be the ith feature vector of the data set, and let $\mathbf{x}_{ij}$ be the jth feature vector of the set in class i. Consequently, $p(y = y_p)$ and $p(\mathbf{x}|y = y_p)$, where $1 \leq p \leq N_c$ can be written as:

$$\begin{aligned} p(y = y_p) &= \frac{J_p}{N}, \\ p(\mathbf{x}|y = y_p) &= \frac{1}{J_p}\sum_{j=1}^{J_p}\mathcal{N}(\mathbf{x} - \mathbf{x}_{pj}, \sigma^2 I), \\ p(\mathbf{x}) &= \frac{1}{N}\sum_{j=1}^{J_p}\mathcal{N}(\mathbf{x} - \mathbf{x}_j, \sigma^2 I). \end{aligned}$$

By the expansion of (3) while using a Parzen estimator with *symmetrical kernel* of width σ, we get the following equation:

$$\mathcal{I}_Q(\mathbf{x}, y) = V_{IN} + V_{ALL} - 2V_{BTW}, \tag{5}$$

where

$$V_{IN} = \sum_y \int_{\mathbf{x}} p(\mathbf{x}, y)^2 d\mathbf{x} = \frac{1}{N^2}\sum_{p=1}^{N_c}\sum_{l=1}^{J_p}\sum_{k=1}^{J_p}\mathcal{N}(\mathbf{x}_{pl} - \mathbf{x}_{pk}, 2\sigma^2 I), \tag{6}$$

$$V_{ALL} = \sum_y \int_{\mathbf{x}} p(\mathbf{x})^2 p(y)^2 d\mathbf{x} = \frac{1}{N^2}\sum_{p=1}^{N_c}\left(\frac{J_p}{N}\right)^2\sum_{l=1}^{N}\sum_{k=1}^{N}\mathcal{N}(\mathbf{x}_l - \mathbf{x}_k, 2\sigma^2 I), \tag{7}$$

$$V_{BTW} = \sum_y \int_{\mathbf{x}} p(\mathbf{x}, y)p(\mathbf{x})p(y)d\mathbf{x} = \frac{1}{N^2}\sum_{p=1}^{N_c}\frac{J_p}{N}\sum_{l=1}^{N}\sum_{k=1}^{J_p}\mathcal{N}(\mathbf{x}_l - \mathbf{x}_{pk}, 2\sigma^2 I). \tag{8}$$

The computational complexity of (5) is comprised of the computational complexity of (6) - (8) and is given in table 1. Furthermore, it is known that the FQMI requires many samples to be accurately computed by Parzen window estimation. Thus, we can assume that when the number of samples N is much greater than their respective dimensionality, that is, $N >> d$, the complexity of V_{ALL}, which is quadratic with respect to N, is dominant for the equation (5).

Table 1. Computational Complexity for terms V_{IN}, V_{ALL}, V_{BTW} [N_c = classes #, N = samples #, J_p = samples # in class p, d = samples' dimension]

FQMI Terms	**Computational Complexity**
V_{IN}	$O(N_c J_p^2 d^2)$
V_{ALL}	$O(N_c N^2 d^2)$
V_{BTW}	$O(N_c N J_p^2 d^2)$

2 Separation Criterions

The standard separation criterion for use in the SFFS algorithm, as proposed by Escalera et al. [4], is the maximization of the Mutual Information between the two created bi-partitions of classes and their respective class labels. That is, in each iteration of the SFFS algorithm two partitions of classes are constructed with labels $\{-1, +1\}$ respectively. As already mentioned, the above procedure is computationaly costly because the FQMI computation in each step of SFFS is applied on all the samples of the considered bi-partitions. We reduce the computational cost if we avoid the computation of FQMI for both of the bi-partitions and apply it only on one of them in each step of SFFS. As can be seen in table 1, another possibility is to avoid computing the term V_{ALL} which is of quadratic complexity with respect to the number of samples N. By discarding the computation of the V_{ALL} term in the FQMI procedure and considering a Fisher like ratio with the available terms V_{IN} and V_{BTW} which are of lower complexity, we can reduce significantly the running time. Finally, we can further reduce the running time if in the Fisher like ratio mentioned, we consider only a representative subset of classes' samples.

Based on these ideas we propose three different variations of the standard criterion $\{C_1, C_2, C_3\}$ which are outlined below:

- **Criterion** C_1: In criterion C_1 we apply the standard FQMI computation only in the current subset of classes that are examined by SFFS in each iteration step. That is, we do not consider in the computation the remaining set of classes that do not belong in the current subset. In this case our goal is to minimize the above measure. In particular, the criterion $J(X, X')$ in the lines 11, 14, 15 of the SFFS algorithm reduces to the criterion $J(X)$. Here, FQMI is evaluated between the subset X and the original class labels of the samples that consist it. The computational complexity of this variation remains quadratic with respect to the number of samples of the group in which the FQMI is evaluated. The evaluation, though, is done using much less data samples and consequently the running time is less than the original approach.
- **Criterion** C_2: In criterion C_2 we consider the maximization of the ratio

$$C_2 = \frac{V_{IN}}{V_{BTW}}$$

where V_{IN} and V_{BTW} are computed as in equations (6) and (8). Here we omit the costly computation of the quantity V_{ALL}. The resulting computational complexity as can be seen from table 1 is quadratic to the number of samples J_p of the binary group, that is $p \in \{-1, +1\}$.

- **Criterion** C_3: The computational cost of FQMI is mostly attributed to the number of samples N. Thus, if we reduce the number of samples we can achieve a drastic reduction of the computational complexity. To this end we can represent each class by only one sample. This sample can be a location estimator such as the *mean* or the *median*. We propose the use of the mean vector as the only representative of each class and the criterion C_2 reduces to minimizing of V_{BTW} where in this case V_{BTW} is given by:

$$V_{BTW} = \frac{1}{N_c^2} \sum_{i=1}^{N_c} \sum_{j=1}^{N_c} \mathcal{N}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j, 2\sigma^2 I)$$

where $\tilde{\mathbf{x}}_i$ is the mean vector of class i.

The new variation has quadratic complexity with respect to the number of classes N_c of the bipartition, since the computation of the mean vectors takes linear time with respect to number of samples in each class J_p.

3 Experimental Results

Datasets. We compared the proposed criteria using eight datasets of the UCI Machine Learning Repository. The characteristics of each dataset can be seen in table 2. All the features of each dataset were scaled to the interval $[-1, +1]$. To evaluate the test error on the different experiments, we used 10-fold cross validation.

Sub-class ECOC configuration. The set of parameters $\theta = \{\theta_p, \theta_s, \theta_i\}$ in the subclass approach were fixed in each dataset to the following values:

- $\theta_p = 0\%$, split the classes if the classifier does not attain zero training error.
- $\theta_s = \frac{|J|}{50}$, minimum number of samples in each constructed cluster, where $|J|$ is the number of features in each dataset.
- $\theta_i = 5\%$, the improvement of the newly constructed binary problems after splitting.

Furthermore, as a clustering method we used the K-means algorithm with the number of clusters $K = 2$. As stated by Escalera et al. [4], the K-means algorithm obtains similar results with other more sophisticated clustering algorithms, such as hierarchical and graph cut clustering, but with much less computational cost.

In the tables 3 and 4 we present the results from our experiments in the UCI datasets using the DECOC and sub-ECOC approaches. In each column we illustrate the corresponding 10 fold cross-validation performance and in the case of the sub-ECOC method the (mean number of rows $\times$ mean number of columns) of the encoding matrices which are formed in each fold.

Table 2. UCI Machine Learning Repository Data Sets Characteristics

Database	Samples	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

Table 3. UCI Repository Experiments for linear SVM C=100

	FQMI		Criterion 1		Criterion 2		Criterion 3	
Database	ECOC	sub-ECOC	ECOC	sub-ECOC	ECOC	sub-ECOC	ECOC	sub-ECOC
Iris	**97.33%**	**97.33%** (3.3 × 2.3)	**97.33%**	**97.33%** (3.3 × 2.3)	**97.33%**	**97.33%** (3.3 × 2.3)	**97.33%**	**97.33%** (3.3 × 2.3)
Ecoli	82.98%	80.71% (10.2 × 10.6)	**84.85%**	**84.85%** (8.2 × 7.2)	78.21%	78.21% (8 × 7)	83.01%	80.63% (8.4 × 7.6)
Wine	96.07%	96.07% (3 × 2)	96.07%	96.07% (3 × 2)	**96.73%**	**96.73%** (3 × 2)	96.07%	96.07% (3 × 2)
Glass	**63.16%**	**66.01%** (13 × 14.3)	60.58%	63.64% (7.1 × 6.1)	61.07%	59.78% (7 × 6)	60.97%	62.85% (9.4 × 8.8)
Thyroid	**96.77%**	**96.77%** (3.3 × 2.6)	**96.77%**	**96.77%** (6 × 7.1)	90.26%	94.89% (5.9 × 7.6)	**96.77%**	**96.77%** (3 × 2)
Vowel	**73.94%**	77.47% (27.2 × 29)	50.91%	52.73% (18.1 × 16.9)	46.26%	45.35% (15.1 × 14)	72.73%	**86.57%** (23.1 × 22)
Balance	**91.7%**	83.56% (54.3 × 64.6)	**91.7%**	**89.31%** (26.4 × 27)	**91.7%**	75.71% (416 × 508)	**91.7%**	88.65% (9.5 × 8.4)
Yeast	**56.6%**	**53.49%** (29.5 × 36.7)	39.36%	39.36% (10 × 9)	42.37%	42.63% (10.2 × 9.2)	47.18%	36.23% (15.7 × 17)

SVM configuration. As a standard classifier for our experiments we used the libsvm [11] implementation of the Support Vector Machine with linear and RBF kernel. For both linear and RBF SVM we fixed the cost parameter C to 100 and for the RBF SVM we fixed the σ parameter to 1.

Table 4. UCI Repository Experiments for RBF SVM C=100, $\sigma = 1$

	FQMI		Criterion 1		Criterion 2		Criterion 3	
Database	ECOC	sub-ECOC	ECOC	sub-ECOC	ECOC	sub-ECOC	ECOC	sub-ECOC
Iris	**96%**	**96%** (3 × 2)	**96%**	**96%** (3 × 2)	**96%**	**96%** (3 × 2)	**96%**	**96 %** (3 × 2)
Ecoli	82.83%	82.56% (13.1 × 16)	**85.10%**	**85.13%** (8.6 × 7.6)	84.08%	84.08% (8.1 × 7.1)	85.04%	85.04% (8.1 × 7.1)
Wine	**97.74%**	**97.74%** (3 × 2)	**97.74%**	**97.74%** (3 × 2)	97.18%	97.18% (3 × 2)	**97.74%**	**97.74%** (3 × 2)
Glass	**69.39%**	**70.78%** (7.9 × 7.6)	**69.39%**	69.39% (6 × 5)	64.77%	64.77% (6 × 5)	68.48%	68.48% (6 × 5)
Thyroid	95.35%	95.35% (3.2 × 2.4)	95.35%	**95.82%** (3.8 × 3.4)	**97.21%**	95.32% (5 × 5.4)	95.35%	95.35% (3 × 2)
Vowel	**99.09%**	**99.09%** (11 × 10)	**99.09%**	**99.09%** (11 × 10)	98.59%	98.59% (11 × 10)	98.99%	98.99% (11 × 10)
Balance	95.04%	95.04% (3 × 2)	95.04%	95.04% (3 × 2)	**95.51%**	**95.51%** (3 × 2)	95.04%	95.04% (3 × 2)
Yeast	**58.6%**	55.44% (27.3 × 33.4)	56.66%	**56.66%** (10 × 9)	54.95%	52.75% (10.5 × 9.5)	56.18%	52.04% (20.7 × 22.1)

From the experiments it is obvious that the proposed criteria attain similar performance in most cases with the FQMI criterion whereas, in terms of computational speed we found that for the tested databases C_1 and C_2 run approximately 4 times faster and criterion C_3 runs approximately 100 times faster. Moreover, FQMI cannot be applied to databases having a great number of

samples. However, the proposed criterion C_3 can be used in very large databases arising in applications such as *Data Mining*.

4 Conclusion

Although FQMI is a quite accurate method for modeling the MI between classes, its computational complexity makes it impractical for real life classification problems. FQMI's inability to address large datasets makes the ECOC - sub-ECOC methods also impractical. As it has been illustrated in our paper, we can substitute FQMI with other MI measures of less computational complexity and attain similar or even in quite a few cases better classification results. These novel MI measures proposed, make the ECOC and sub-ECOC methods applicable in large real-life datasets.

References

1. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multi-class to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research 1, 113–141 (2002)
2. Dietterich, T.G., Bakiri, G.: Solving multi-class learning problems via error-correcting output codes. Journal of Machine Learning Research 2, 263–282 (1995)
3. Kong, E., Dietterich., T.: Error-correcting output coding corrects bias and variance. In: Proc. 12th Intl Conf. Machine Learning, pp. 313–321 (1995)
4. Escalera, S., Tax, D.M., Pujol, O., Radeva, P., Duin, R.P.: Subclass problem-dependent design for error-correcting output codes. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(6), 1041–1054 (2008)
5. Torkkola, K.: Feature extraction by non-parametric mutual information maximization. Journal of Machine Learning Research 3, 1415–1438 (2003)
6. Asuncion, A., Newman, D.: Uci machine learning repository (2007)
7. Escalera, S., Pujol, O., Radeva, P.: Loss-weighted decoding for error-correcting output coding. In: Proc. Int'l Conf. Computer Vision Theory and Applications, June 2008, vol. 2, pp. 117–122 (2008)
8. Pujol, O., Radeva, P., Vitria, J.: Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. IEEE Transactions on Pattern Analysis and Machine Intelligence 6, 1001–1007 (2006)
9. Pudil, P., Ferri, F., Novovicova, J., Kittler., J.: Floating search methods for feature selection with non-monotonic criterion functions. In: Proc. Int'l Conf. Pattern Recognition, March 1994, vol. 3, pp. 279–283 (1994)
10. Kapur, J., Kesavan, H.: Entropy Optimization principles with Applications (1992)
11. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines (2001)