BeamECOC: A Local Search for the Optimization of the ECOC Matrix

Cemre Zor*, Berrin Yanikoglu†, Erinc Merdivan†, Terry Windeatt*, Josef Kittler*, Ethem Alpaydin‡
*Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford, United Kingdom, GU2 7XH
{c.zor, t.windeatt, j.kittler} @surrey.ac.uk
†Faculty of Engineering and Natural Sciences, Sabanci University, Tuzla, Istanbul, Turkey, 34956
{berrin, merdivan}@sabanciuniv.edu
‡Department of Computer Engineering, Bogazici University, Bebek, Istanbul, Turkey, 34342
alpaydin@boun.edu.tr

***Abstract*—Error Correcting Output Coding (ECOC) is a multi-class classification technique in which multiple binary classifiers are trained according to a preset code matrix such that each one learns a separate dichotomy of the classes. While ECOC is one of the best solutions for multi-class problems, one issue which makes it suboptimal is that the training of the base classifiers is done independently of the generation of the code matrix.**

In this paper, we propose to modify a given ECOC matrix to improve its performance by reducing this decoupling. The proposed algorithm uses beam search to iteratively modify the original matrix, using validation accuracy as a guide. It does not involve further training of the classifiers and can be applied to any ECOC matrix.

We evaluate the accuracy of the proposed algorithm (BeamECOC) using 10-fold cross-validation experiments on 6 UCI datasets, using random code matrices of different sizes, and base classifiers of different strengths. Compared to the random ECOC approach, BeamECOC increases the average cross-validation accuracy in 83.3% of the experimental settings involving all datasets, and gives better results than the state-of-the-art in 75% of the scenarios. By employing BeamECOC, it is also possible to reduce the number of columns of a random matrix down to 13% and still obtain comparable or even better results at times.

***Index Terms*—ECOC, Error correcting output codes, ensemble, learning, beam search**

I. Introduction

In many pattern recognition problems, an ensemble of classifiers is shown to achieve a higher expected generalization ability than the individual classifiers (base classifiers) it is composed of. The classifier combination methods can be as simple as taking a vote between individual classifiers trained to solve the given problem, or in more complex ways, where the individual classifiers are trained to compensate for weaknesses of previous classifiers. Over the last decade, considerable research has been conducted on ensembles to investigate different methodologies for their construction and to seek their theoretical underpinning [1].

Error Correcting Output Coding (ECOC) is an ensemble classification technique proposed specially for multi-class classification problems [2]. In ECOC, a number of binary classifiers are trained such that each one is assigned a separate dichotomy of the classes, which is defined by a given ECOC matrix. The phase where the ECOC matrix is constructed is called the *encoding/design* stage, and there are various data dependent and independent approaches for encoding the ECOC matrix, which will be visited in Section II.

Consider a problem with K classes $\{c_1 \dots c_K\}$ and L base classifiers $\{h_1 \dots h_L\}$, and an encoded ECOC matrix M of size $K \times L$. In M, a particular element $M_{ij} \epsilon \{+1, -1\}$ indicates the desired label for class c_i to be used in training the base classifier h_j. In Fig. 1, an example ECOC matrix is given for a 5 class problem to be solved using 6 base classifiers. Here, for instance, the base classifier h_1 is assigned the task of learning samples from classes c_1, c_2, c_3 as positive instances and c_4, c_5 as negative. The i^{th} row of M, denoted as M_i, is the *codeword* indicating the desired output for class c_i.

Fig. 1: A sample code matrix for a 5-class classification problem with 6 classifiers.

	h_1	h_2	h_3	h_4	h_5	h_6
c_1	+1	+1	+1	-1	-1	-1
c_2	+1	-1	-1	+1	-1	-1
c_3	+1	+1	-1	-1	-1	-1
c_4	-1	-1	-1	+1	+1	-1
c_5	-1	+1	+1	-1	+1	+1

In the *decoding/testing* stage, a given test sample x is firstly classified by each base classifier, to obtain the output vector $y = [y_1, ..., y_L]$ where y_j is the hard or soft output of the classifier h_j for x. Then, for all i, the distance between y and the codeword M_i of class c_i is computed by using a metric such as the Hamming, Manhattan or Euclidean distance. The class c_k, for which the minimum distance is obtained, is chosen as the estimated class label, such that

$$k = argmin_{i=1...K}\ d(y, M_i) \tag{1}$$

Choosing the closest codeword enables the system to correct some of the mistakes of the base classifiers, hence providing some error correction. Specifically, while employing Hamming Distance (HD), if the minimum HD between any pair of codewords is d, then up to $\lfloor (d-1)/2 \rfloor$ single bit errors can be corrected.

It should be mentioned here that the *ternary* ECOC is later on proposed by Allwein et al in [3] in order to simplify the task of the dichotomizers. In this encoding, M_{ij} can take values from $\{+1, 0-1\}$, where the zero element is used to indicate the classes that are to be taken out of the consideration of a base classifier and not used in its training. During the decoding of the ternary matrices, there are many distance metrics suggested for properly handling the zero entries [4]. A straight-forward decoding methodology would need to ignore the differences in the zero entries, such that the distance metric $d(y, M_i)$ used in Eq. 1 becomes

$$d(y, M_i) = \frac{\sum_{j=1..L} |M_{ij}||y_j - M_{ij}|}{\sum_{j=1..L} |M_{ij}|} \quad (2)$$

where the differences in non-zero entries that are summed in the numerator are normalized by the number of non-zero entries in M_i. In case the output has the same distance to two separate code words, normalization gives more weight to the codeword having a larger number of non-zero entries.

Although there are various ways to design (encode) and test (decode) ECOC matrices, there has been relatively less interest in the implementations of efficient update methodologies of already encoded matrices. The rationale for an update strategy is rooted in the decoupling of the ECOC matrix design and base classifier training: Although the sub-problems assigned to base classifiers are generally significantly simpler compared to the overall classification problem, it is shown that some of the base classifiers might still end up with accuracies close to 0% for some classes in many problems and experimental settings as analysed in [5].

In this study, we propose a novel problem-independent update methodology, namely BeamECOC, which increases the ensemble accuracy by modifying the preset ECOC matrix so as to better match the trained base classifiers, using a local iterative search technique by employing a validation set. Experimental evidence demonstrates that the method improves classification accuracy in 83.3% of the experimental settings, and gives better results than the state-of-the-art in 75% of the time. BeamECOC can also help reduce the size of an ensemble down to 13% while still providing comparable or, in some cases, even better results.

The organization of the paper is as follows. After reviewing previous work carried out in this area in Section II, the proposed optimization is described in Section III. The experimental results are given in Section IV, and finally, an evaluation of the proposed scheme together with suggestions of future research directions are provided in Section V.

II. Previous Work

Inspired by the concept of the ECOC method, many variations for the encoding and decoding steps have been suggested in the literature. For the encoding of the ECOC matrix, there are some commonly used problem-independent techniques such as one-versus-all, one-versus-one, dense random and sparse random where the base classifiers are assigned the tasks of separating one class from the rest, one class from another, or random dichotomies with few or many zeros, respectively [3]. In addition, there is the computationally expensive alternative of exhaustive codes, but they do not guarantee the best performance.

On the other hand, the idea in data-dependent ECOC designs is to encode matrices by making use of the input training data in order to end up with base classifiers which are meaningful within a given domain. Pujol et al. in [6] have proposed Discriminant ECOC (DECOC), which aims to create superclasses that maximize the discriminability within a given problem using mutual information. The superclasses, which are ranked on a tree structure using floating search, are then embedded into ECOC. Later on in Subclass Problem-Dependent Design [7], DECOC has been modified in order to split the original set of classes into more distinguishable subclasses. A further extension of DECOC, namely Forest-ECOC has then been proposed in [8] where the aim is to use multiple trees while splitting the original set of classes into distinguishable ones.

It is important to mention here that although problem-dependent coding approaches are successful, it has been theoretically and experimentally proven that randomly generated long or deterministic equidistant code matrices deliver close to optimum performance when used with strong base classifiers [9], [10].

As for the decoding of ECOC, there are many strategies used apart from standard decoding technique based on the Hamming or Euclidean distance. Standard decoding can be perceived as the minimization of the distance between a multidimensional codeword vector and target vertices of the multidimensional ECOC hypercube. However, if patterns belonging to a class tend to form up a cluster, then the centroid of this cluster might be a more suitable target vector for the class of interest than the already existing ECOC codewords. In other words, each class can end up re-defining its own representative. This decoding method is called Centroid of Classes [9]. In Loss-based Decoding [3], the aim is to find the class label among the set of possible labels that minimizes the total loss computed using all base classifier decisions for a given test instance, where the loss function is dependent on the 'margin' of this instance. Finally in Probabilistic-based Decoding [11], Passerini et al. suggest making use of class conditional probabilities derived from base classifiers for decoding. Detailed descriptions of the existing decoding methodologies have been presented in [12] and [4]. Furthermore in [4], in order to overcome the problems arising while decoding the zero symbol, novel decoding strategies to be used with ternary matrices have been proposed.

In addition to the general work aiming to improve the ECOC approach through better encoding or decoding strategies, there has been little work done on the update of the ECOC matrix as a post-processing step to further improve the accuracy. In an early work aiming addressing the problem of joint optimization of the base classifier training and the ECOC encoding [13], Alpaydin and Mayoraz train a multilayer perceptron to update the ECOC matrix, allowing small modifications from the original.

In ECOC-Optimizing Node Embedding (ECOC-One) proposed in [14], the purpose is to extend any ECOC matrix

by adding a few dichotomizers based on a discriminability criterion defined on the data domain. Iteratively, new base classifiers are added to the coding design, by minimizing the confusion matrix of classes using a validation set. As a result, the HD between the classes that most overlap is aimed to be increased, and the generalization performance to be improved. A further contribution of ECOC-One is the use of a weighting strategy to define the relevance of the base classifiers.

Later on in [15], Escalera et al. propose to update a one-versus-one coding matrix in a problem-dependent way, resulting in an increase in the generalization capability of the system. In this approach, the original code matrix has many zero entries and their method consists of changing a 0 entry of a dichotomizer to +1 or -1, if the corresponding class happens to be correctly classified by this dichotomizer. In a more recent work, Zhong et al. address the idea of code optimization by formulating a framework that takes into account the misclassification errors of test instances using SVMs as base classifiers, along with the Hamming distance between different columns [16]. In this method called JointECOC, it has been reported that the problem is NP-complete whose exact solution is computationally intractable and an approximate solution can be obtained after relaxing some constraints. Finally, Bautista et al. optimize the ECOC matrix by applying methods which are genetically inspired, such as mutation and cross-over [17].

The work proposed in this study is based on updating the preset ECOC matrix using operations of flip (changing +1 bits to -1 and vice versa) and zero-ing (changing +1/-1 to 0), by utilizing a local optimization technique. For this purpose, beam search [18] is employed, as it is a well-known search algorithm that limits the number of nodes expanded at each level of a breadth-first search in order to deal with time and space issues in exponential search problems. The proposed method, namely BeamECOC, is described in Section III.

Algorithm 1 BeamECOC

Input: Code matrix M; trained base classifiers H; thresholds α, β; beam width k
Output: Modified code matrix M
Calculate the accuracy matrix A according to M and H;
$Beam$ = $\{M\}$; ▷ Start search from with the preset code matrix
while $Beam \neq \emptyset$ **do** ▷ Expand as long as $Beam$ is non-empty
  $NewNodes = \emptyset$;
  for each code matrix m in $Beam$ **do**
    for each possible update location (i, j) **do**
      $M' \leftarrow m$; ▷ Create a child node
      if $A_{ij} < \beta$ **then** ▷ Apply appropriate update
        Flip M'_{ij};
      else if $\beta \leq A_{ij} < \alpha$ **then**
        Zero M'_{ij};
      end if
      $\Delta gain \leftarrow$Accuracy[M']$-$Accuracy[m]; ▷ Measure improvement on validation set
      if $gain \geq 0$ **then**
        $NewNodes \leftarrow NewNodes \cup M'$; ▷ If gain is positive, add M' to the new level
      end if
    end for
  end for
  $Beam \leftarrow$ the k best code matrices from $NewNodes$ (last level)
end while
Return best code matrix found so far

III. Proposed Method

Consider an ECOC matrix M and a set of base classifiers that are trained according to this code matrix. If one measures the accuracies of the trained classifiers on a validation set separately for each class, we obtain what we call the accuracy matrix, A, which is of the same size as M. To be precise, A_{ij} is measured as the proportion of the samples in class c_i that are correctly classified by h_j according to the target value specified by M_{ij}. In other words, M_{ij} indicates the target and A_{ij} indicates the accuracy of classifier h_j for class c_i. Our approach has originated from the consideration of what the A matrix may look like after training; how many of its elements may have low values corresponding to bad performance, and what it could tell us about the final solution.

The rationale can be explained using a simple example. Assume that a classifier h_j is fully wrong in classifying a particular class c_i when the target for this class is -1. I.e. $M_{ij} = -1$ and $A_{ij} = 0$. In this situation, changing the M_{ij} value from -1 to +1 corresponds to matching the code matrix to the trained classifier h_j, as the classifier could not do this during the actual training. This modification results in changing A_{ij} to 100% while leaving other entries in A and M unchanged.

It is important to mention here that although the accuracy of h_j for c_i increases in the updated scenario, the overall ECOC classification accuracy may still decrease. This is mainly due to the potential decrease in Hamming distance between class c_i and some of the remaining classes after the update, which then brings about a lowered error-correcting capability. In order to weight the overall effect of a codeword change such as the one given in this example, we propose an algorithm, namely BeamECOC, which modifies the code matrix M using beam search. In each iteration, the accuracy matrix A guides the search to potential changes, while the overall effects of these changes are measured on the validation set to select the best subset that form the beam. Note that the base classifiers remain unchanged in this process.

We start by creating a search tree with the preset code matrix M assigned to the root. At each step of the search, a parent node is expanded to generate child nodes, each of which differs from the parent in a single entry that corresponds to a low A value. Specifically, each child node is assigned a new code matrix, which is obtained by flipping an entry M_{ij} if $A_{ij} < \beta$, or making M_{ij} zero if $\beta \leq A_{ij} < \alpha$. The aim here is to: 1) Flip the entry if it belongs to a class whose patterns are mostly misclassified by the associated base classifier 2) Take the entry out of the consideration of the associated base classifier (make it zero) if it belongs to a class whose patters mainly lay at the confusion region. Hence, in BeamECOC, we suggest using β and α values of $\sim$0.4 and $\sim$0.6, respectively. Note that the number of child nodes derived from a parent node is as many

as the number of entries with corresponding A values less than α.

Beam search is a constrained form of breadth-first search, in which at each step of the search, only up to k nodes are selected for expansion, where k is the beam width. Hence in BeamECOC, at a given level, the selection of the best k out of all constructed child nodes is accomplished according to the validation set accuracy. The selected best k nodes then re-enter the expansion procedure, and the search continues until no further improvement is obtained in the validation set accuracy or the node expansion is complete. Note that if k was unlimited, beam search would be equal to breadth-first search.

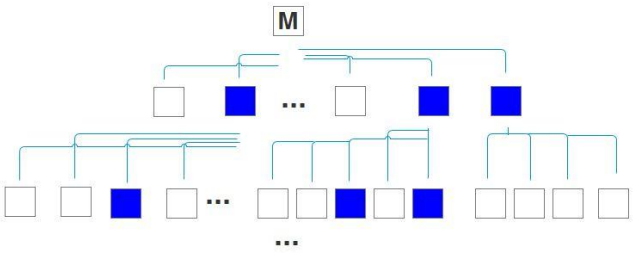

Fig. 2: Illustration of beam search with a beam width of 3.

In Fig. 2, an example beam search is illustrated for $k = 3$. The squares in the first level of the tree are the new code matrices that differ from the original code matrix M in a single entry. Then, these new code matrices are evaluated for their performance on the validation set and the top 3 ones that show the best improvement over M (filled squares in Fig. 2) are expanded in the next iteration of the search. By considering the validation accuracy in this process, we expect the method to take care of the error correction capacity, and therefore carry out updates without causing any degradation. The pseudo-code for the algorithm is given Algorithm 1.

Although BeamECOC updating strategy can be applied to any ECOC framework, in this study we focus on its application to the random code matrices. It has been mentioned in Section II that randomly generated long matrices are theoretically and experimentally proven to give close to optimum performance when they are used with strong base classifiers [9], [10]. Hence, by varying the base classifier parameters and/or the number of columns, it is possible to capture and analyse the effect of BeamECOC on ensembles of different strength.

IV. Experiments

In this section, we experimentally analyze the effect of the proposed update strategy, BeamECOC, on randomly generated code matrices of varying lengths (10 or 75 columns) when used with base classifiers trained as Multilayer Perceptrons (MLPs) of different strengths. The training is done using the Levenberg-Marquart algorithm, for different durations varying between 2 and 15 epochs, and using a number of nodes between 2 and 8.

The aim is to show that BeamECOC can boost the generalization performance of random code matrices, even when they give close to optimal performance as a result of consisting of high number of columns and having associated strong base classifiers. Moreover, the updated random matrices are expected to perform better than the state-of-the-art (SOA) algorithms in various scenarios, including the cases where they are trained with weaker classifiers than those of the SOA.

	# Training Samples	#Test Samples	#Attributes	#Classes
Dermatology	358	-	34	6
Glass Identif.	214	-	9	6
Satellite Image	4435	2000	36	6
Vehicle	946	-	18	4
Yeast	1484	-	8	10
Optical Digits	3823	1797	64	10

TABLE I: Summary of the UCI MLR datasets used in the performance evaluation

The SOA methods investigated include one-versus-all, one-versus-one, ECOC-One, DECOC and Forest-ECOC approaches. For the sake of simplicity, we use simple ternary HD decoding (See Eq 2), during the decoding of the random and BeamECOC-updated random matrices. However, Loss-weighted Decoding (based on a linear loss function) has been utilized instead of HD in the decoding of SOA, as it has been shown to reveal better results when used with these algorithms [4].

BeamECOC is analysed on 6 UCI Machine Learning Repository datasets [19] summarized in Table I. For the datasets which include specific test sets, the validation set is built from within the input training set. As for the sets that do not have a designated test set, 10-fold cross-validation is used to create the training and test sets, and a further random split of the training samples into training and validation sets is carried out. The size of the validation set has experimentally been evaluated and is selected to be set as equal to that of the training. All experiments have been repeated using 10 independent runs and the results are averaged.

In Fig. 3, a comparison of BeamECOC applied to random matrices with a high number of columns (75) against the SOA is shown for weak base classifiers trained with 2 nodes and 2 epochs (a), and for strong classifiers trained with 8 nodes and 15 epochs (b). The comparison also includes the random coding before the application of the BeamECOC update. For each dataset, the best generalization performances obtained are indicated by bold whereas the second best are underlined.

Analysing both Fig. 3(a) and Fig. 3(b), it can be observed that BeamECOC achieves the best overall performance in 75% of the scenarios, and is among the top two most successful algorithms 82% of the time. Moreover, the application of BeamECOC over random coding matrices degrades the performance only in 1 out of 12 settings, and improves it in 83.3% of the cases.

At this point, it has to be borne in mind that the higher the number of columns of a random matrix is and/or the stronger its base classifiers get, the better its corresponding generalization performance becomes. This fact shows its effect as a reduction in the relative accuracy gain between the updated

approach and random coding when the results in Fig. 3(b) are compared to those in Fig. 3(a). This may further be confirmed by looking at a detailed comparison of the BeamECOC with random coding on two example datasets given in Fig. 4 for 2 nodes (a) and 8 nodes (b), and for a varying number of columns and epochs. Here, the best performances are indicated in bold and the statistically significant differences between the two methods based on t-test are underlined. Although in 91% of the scenarios the BeamECOC update shows improvement over random coding, as the ensemble gets stronger, e.g. for 75 columns and 15 epochs, the relative accuracy gain becomes less significant as expected. As for the number of the average flip and zero operations, we see that for weaker ensembles, e.g. for those trained with 2 nodes / 2 epochs / 10 columns, the total number of operations can be as high as 16.7%. Noting that even in stronger ensembles, e.g. in those associated with 8 nodes / 2 epochs / 75 columns, there might be as many as 13% operations encountered, proves that there is room for improvement even for these ensembles.

2Nodes/2Epochs	Derma.	Glass	Satellite	Vehicle	Yeast	Optdigits
OneVsAll	60.60	22.06	54.40	37.85	10.71	38.06
OneVsOne	76.60	49.77	66.04	53.81	45.35	70.77
ECOCONE	66.67	32.78	60.31	43.20	16.50	38.40
DECOC	69.27	40.71	62.99	52.51	36.32	50.94
Forest	77.90	43.40	70.05	52.03	41.11	65.20
Random-75C	82.47	48.17	62.88	52.19	31.37	56.36
BeamECOC-75C	**94.99**	**56.93**	**81.06**	**66.19**	**49.05**	**79.06**

(a) 2 nodes / 2 epochs

8Nodes/15Epochs	Derma.	Glass	Satellite	Vehicle	Yeast	Optdigits
OneVsAll	90.52	52.87	86.16	76.13	40.44	92.08
OneVsOne	94.14	64.94	87.69	81.33	**59.10**	96.97
ECOCONE	96.09	55.35	86.94	78.37	41.44	92.97
DECOC	94.11	58.63	78.45	80.25	50.06	93.83
Forest	95.50	67.45	85.52	**82.28**	58.41	95.24
Random-75C	95.27	68.97	87.28	80.39	54.65	**97.36**
BeamECOC-75C	**96.66**	**69.81**	**88.28**	80.39	55.86	97.28

(b) 8 nodes / 15 epochs

Fig. 3: Comparison of BeamECOC with state-of-the-art algorithms using weak (a) and strong base classifiers (b). Given in terms of accuracy (%).

A follow-up question to the above analysis would be "If *long enough* random matrices engaging *strong enough* base classifiers give close to optimum performance, why would there be a need for an update strategy?". It has to be mentioned here that, although the error rate of a random coding gets close to optimum exponentially with the increasing number of columns (and hence a convergence to 'close to ideal' performance is achieved), the number of columns required for an *optimum* performance would have to go to infinity [9]. Moreover, to achieve the optimum performance, base classifiers used in this ensemble would theoretically need to be the Bayes classifiers. This means, despite *long enough* number of columns and *strong enough* base classifiers, there will always be room for

2 Nodes	10Col/2E	10Col/15E	25Col/2E	25Col/15E	75Col/2E	75Col/15E
Derma.						
Random	58.67±9.9	76.51±12.5	58.72±11.1	85.25±13.8	82.47±9.3	93.88±5.3
BeamECOC	**84.01±8.9**	**90.78±3.2**	**85.21±4.5**	**93.32±3.6**	**94.99±4.3**	**94.99±2.8**
Avg. Flipped	14.30	7.00	15.10	4.60	7.40	3.20
Avg. Zeroed	2.10	0.10	1.30	1.00	1.60	0.30
Vehicle						
Random	33.30±9.8	58.57±14.6	39.24±8.8	69.18±10.2	52.19±12.5	76.25±4.2
BeamECOC	**50.46±10.8**	**71.78±5.8**	**52.96±10.1**	**76.13±4.8**	**66.19±4.3**	**77.91±4.1**
Avg. Flipped	13.20	9.70	9.70	6.00	9.00	4.80
Avg. Zeroed	3.50	2.70	4.00	4.20	4.00	3.50

(a) 2 nodes

8 Nodes	10Col/2E	10Col/15E	25Col/2E	25Col/15E	75Col/2E	75Col/15E
Derma.						
Random	84.07±7.5	96.67±2.5	77.13±7.1	92.47±3.9	74.55±9.9	95.27±3.1
BeamECOC	**91.05±3.4**	**96.94±2.7**	**95.55±2.6**	**94.99±2.1**	**94.99±3.6**	**96.66±2.1**
Avg. Flipped	10.60	3.50	11.60	4.60	12.00	4.60
Avg. Zeroed	1.70	0.40	0.40	0.00	1.00	0.00
Vehicle						
Random	49.51±6.2	76.73±4.1	63.60±4.3	**79.69±4.6**	65.80±7.0	**80.39±3.5**
BeamECOC	**63.36±6.6**	**77.32±5.6**	**68.43±2.7**	78.86±4.4	**69.73±4.3**	**80.39±3.3**
Avg. Flipped	8.50	3.60	4.30	3.50	3.50	3.00
Avg. Zeroed	5.30	4.90	2.60	2.30	4.10	2.60

(b) 8 nodes

Fig. 4: Detailed comparison of BeamECOC with random coding, for 2 nodes (a), 8 nodes (b). Given in terms of accuracy with std. deviation, and averaged number of flips and zeros (%).

8Nodes/15Epochs	Derma.	Glass	Satellite	Vehicle	Yeast	Optdigits
Random-10C	96.67	65.99	85.68	76.73	52.36	94.43
BeamECOC-10C	**96.94**	**68.29**	**87.78**	77.32	54.45	94.98

Fig. 5: BeamECOC for 10 column random code matrices, used with 8 nodes / 15 epochs. Given in terms of accuracy (%).

improvement in the random coding matrix.

Yet another advantage brought about by BeamECOC is the possibility to employ simpler ensembles to achieve a better generalization performance than the original random coding. In Fig. 5, experiments given in Fig. 3 (b) for 8 nodes and 15 epochs are repeated for random coding matrices of 10 columns (instead of the previously used 75 columns) and their BeamECOC-updates. Comparing the two settings, the cases where BeamECOC on 10 column codes achieve results which are not significantly different from those of the original coding with 75 columns are underlined. In other words, in 4 out of 6 datasets, it is possible to shrink the training complexity to 13% by reducing the number of random columns from 75 to 10 followed by a BeamECOC update, and still obtain significantly indifferent and/or better results. On the other hand, when compared to the SOA, the cases where the 10 column updated matrices obtain better results are indicated in bold: For half of the datasets, it is still possible to achieve better performances than the SOA even while using shorter random matrices updated by BeamECOC. Using the same rationale, these observations further suggest the possible use of weaker base classifiers with longer matrices to obtain significantly

indifferent/better results compared to those (of the same length) trained with strong base classifiers. These findings are of importance for systems subject to time and complexity constraints as well as high generalization performance.

V. Summary

In this paper, we have presented a novel update scheme applicable to already trained ECOC matrices, and analysed its use with random coding matrices under different settings while varying ensemble strength. The findings suggest that the proposed algorithm improves performance for 83.3% of the cases, and achieves the best overall generalization capability compared to the state-of-the-art methodologies in 75% of the scenarios. In our experiments, up to 16.7% of the code matrix elements were modified as a result of the update processes. It has also been shown that the use of the update methodology brings about a reduction in ensemble complexity without significantly degrading performance, which is an important feature to be employed in systems with time/complexity constraints.

Future work will investigate the update scheme in conjunction with different coding schemes rather than random, and analysing the complexity/accuracy trade-off in more comprehensive scenarios including real-life datasets. Different decoding strategies taking care of the zero symbol are also to be analysed so as to achieve even better generalization performances.

Acknowledgements

This work was partially carried out as part of EPSRC project "Signal processing in a networked battlespace" under contract EP/K014307/1 and the MOD University Defence Research Collaboration (UDRC) in Signal Processing.

References

[1] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[2] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.

[3] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *J. Mach. Learn. Res.*, vol. 1, pp. 113–141, September 2001.

[4] S. Escalera, O. Pujol, and P. Radeva, "On the decoding process in ternary error-correcting output codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 120–134, January 2010.

[5] C. Zor, B. Yanikoglu, T. Windeatt, and E. Alpaydin, "FLIP-ECOC: a greedy optimization of the ECOC matrix," in *Proceedings of the 25th International Symposium on Computer and Information Sciences (ISCIS 2010)*. Springer, 2010, pp. 149 – 154.

[6] O. Pujol, P. Radeva, and J. Vitria, "Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 1007–1012, 2006.

[7] S. Escalera, D. M. J. Tax, O. Pujol, P. Radeva, and R. P. W. Duin, "Subclass problem-dependent design for error-correcting output codes," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1041–1054, 2008.

[8] S. Escalera, O. Pujol, and P. Radeva, "Boosted landmarks of contextual descriptors and forest-ECOC: A novel framework to detect and classify objects in cluttered scenes," *Pattern Recognition Letters*, vol. 28, no. 13, pp. 1759–1768, 2007.

[9] G. M. James, "Majority vote classifiers: Theory and applications." Ph.D. dissertation, Department of Statistics, University of Standford, 1993.

[10] G. M. James and T. Hastie, "The error coding method and PICT's," *Computational and Graphical Statistics*, vol. 7, no. 3, pp. 377 – 387, 1998.

[11] A. Passerini, M. Pontil, and P. Frasconi, "New results on error correcting output codes of kernel machines," *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 45–54, 2004.

[12] T. Windeatt and R. Ghaderi, "Coding and decoding strategies for multi-class learning problems," *Information Fusion*, vol. 4, no. 1, pp. 11 – 21, 2003.

[13] E. Alpaydin and E. Mayoraz, "Learning error-correcting output codes from data," in *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN 1999).*, vol. 2, 1999, pp. 743 –748.

[14] S. Escalera and O. Pujol, "Ecoc-ONE: A novel coding and decoding strategy," in *International Conference on Pattern Recognition (3)*, 2006, pp. 578–581.

[15] S. Escalera, O. Pujol, and P. Radeva, "Recoding error-correcting output codes," in *Proceedings of the 8th International Workshop on Multiple Classifier Systems*, ser. MCS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 11–21.

[16] G. Zhong, K. Huang, and C.-L. Liu, "Learning ECOC and dichotomizers jointly from data," in *ICONIP (1)*, 2010, pp. 494–502.

[17] M. Á. Bautista, S. Escalera, X. Baró, P. Radeva, J. Vitrià, and O. Pujol, "Minimal design of error-correcting output codes," *Pattern Recognition Letters*, vol. 33, no. 6, pp. 693–702, 2012.

[18] W. Zhang, *State-space search - algorithms, complexity, extensions, and applications*. Springer, 1999.

[19] A. Asuncion and D. J. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html