IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Empirical Study On Error Correcting Output Code Based On Multiclass Classification

Ms. Devangini Dave[1], Prof. M. Samvatsar[2], Prof. P. K. Bhanodia[3]
devangipdave@yahoo.co.in

Abstract

A common way to address a multi-class classification problem is to design a model that consists of hand picked binary classifiers and to combine them so as to solve the problem. Error-Correcting Output Codes (ECOC) is one such framework that deals with multi-class classification problems. Recent works in the ECOC domain has shown promising results demonstrating improved performance. Therefore, ECOC framework is a powerful tool to deal with multi-class classification problems. The error correcting ability improves and enhances the generalization ability of the base classifiers. This paper introduces state-of-the-art coding (one-versus-one, one-versus-all, dense random, sparse random, DECOC, forest-ECOC, and ECOC-ONE) and decoding designs (hamming, Euclidean, inverse hamming, laplacian, β-density, attenuated, loss-based, probabilistic kernel-based, and loss weighted) perspectives along with empirical study of ECOC following comparison of various ECOC methods in the above context. Towards the end, our paper consolidates details relating to comparison of various classification methods with Error Correcting Output Code method available in wake, after carrying out experiments with weak tool as a final supplement to our studies.

Index Terms—Coding, Decoding, Error Correcting Output Codes, Multiclass Classification.
.

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 < x_i; C (x_i) >. C(x) is usually a set of discrete labels. For example, in face detection, C(x) is a binary function C(x) ∊ {face, nonface}, in optical digit recognition C(x) ∊ {0,…, 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 [4] 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 [5], resulting in an N_c (N_c - 1)/2 set of binary problems. Dietterich and Bakiri [7] 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 sub problems, where n is the error correcting output code length n ε {N_c,…,∞}. Then, the output of all classifiers must be combined—traditionally using Hamming distance.

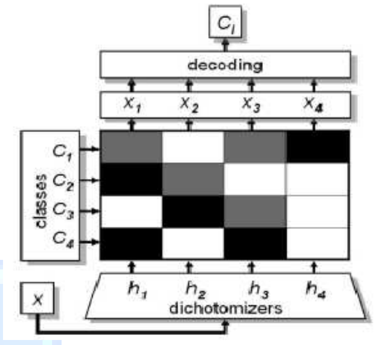
The approach of Dietterich and Bakiri was improved by Allwein et al. [6] by introducing an uncertainty value in the ECOC design and exploring alternatives for mixing the resulting outputs of the classifiers. In particular, they introduced loss-based decoding as a way of merging the classifiers. Recently, 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 [8], [9]. 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.

Error Correcting Output Codes

Given a set of N_c classes, the basis of the ECOC framework consists of designing a codeword for each of the classes.

1) These code words encode the membership information of each class for a given binary problem.
2) Arranging the code words as rows of a matrix, we obtain "a coding matrix M_c", where $M_c \in \{-1, 0, +1\}^{N_c \times n}$, being n the length of the code words codifying each classes.
3) From the point of view of learning, M_c is constructed by considering n binary problems each one corresponding to a column of the matrix M_c.
4) Each of these binary problems splits the set of classes in two partitions (coded by +1 or -1 in M_c according to their class set membership or 0 if the class is not considered by the current binary problem).
5) Then at the decoding step, applying the n trained binary classifiers, a code is obtained for each data point in the test set.
6) This code is compared to the base code words of each class defined in the matrix M_c, and the data point is assigned to the class with the closest codeword.

   Below figure show ECOC coding design for a 4-class problem. White, black and grey positions corresponds to the symbols +1, -1 and 0, respectively. Once the four binary problems are learnt, at the decoding step a new test sample X is tested by the n classifiers. Then the new codeword x={x_1,…, xn} is compared with the class code words {C_1,…, C4}, classifying the new sample by the class C_i which codeword minimizes the decoding measure.



Coding Design

Here the ECOC coding design covers the state-of-the art of coding strategies, mainly divided in two main groups: problem-independent approaches, which do not take into account the distribution of the data to define the coding matrix, and the problem-dependent designs, where information of the particular domain is used to guide the coding design.

Problem-Independent ECOC Coding Designs

- One-versus-all (Rifkin and Klautau, 2004): N_c dichotomizers are learnt for *Nc* classes, where each one splits one class from the rest of classes.
- One-versus-one (Nilsson, 1965): $n = N_c(N_c - 1)/2$ dichotomizers are learnt for N_c classes, splitting each possible pair of classes.
- Dense Random (Allwein et al., 2002): $n = 10 \cdot \log N_c$ dichotomizers are suggested to be learnt for N_c classes, where $P(-1) = 1-P(+1)$, being $P(-1)$ and $P(+1)$ the probability of the symbols -1 and +1 to appear, respectively. Then, from a set of defined random matrices, the one which maximizes a decoding measure among all possible rows of M_c is selected.
- Sparse Random (Escalera et al., 2009): $n = 15 \cdot \log N_c$ dichotomizers are suggested to be learnt for N_c classes, where $P(0) = 1-P(-1) - P(+1)$, defining a set of random matrices M_c and selecting the one which maximizes a decoding measure among all possible rows of M_c.

Problem-Dependent ECOC Coding Designs

- DECOC (Pujol et al., 2006): problem-dependent design that uses $n = N_c - 1$ dichotomizers. The partitions of the problem are learnt by means of a binary tree structure using exhaustive search or a SFFS criterion.

Finally, each internal node of the tree is embedded as a column in M_c.

- Forest-ECOC (Escalera et al., 2007): problem-dependent design that uses $n = (N_c - 1) \cdot T$ dichotomizers, where T stands for the number of binary tree structures to be embedded. This approach extends the variability of the classifiers of the DECOC design by including extra dichotomizers.
- ECOC-ONE (Pujol et al., 2008): problem-dependent design that uses $n = 2 \cdot N_c$ suggested dichotomizers. A validation sub-set is used to extend any initial matrix M_c and to increase its generalization by including new dichotomizers that focus on difficult to split classes.

Decoding Design

The notation used refers to that used in (Escalera et al., 2008):

- Hamming decoding:
$HD(x, y_i) = \sum_{j=1}^{n}(1 - sign(x^j y_i^j))/2$, being x a test codeword and yi a codeword from M_c corresponding to class C_i.
- Inverse Hamming decoding: $IHD(x, yi)$ = max ($\Delta^{-1} D^T$), where $\Delta(i_1, i_2) = HD(y_{i1}, y_{i2})$, and D is the vector of Hamming decoding values of the test codeword x for each of the base code words y_i.
- Euclidean decoding:
$ED(x, y_i) = \sqrt{\sum_{j=1}^{n}(x_j - y_j)^2}$
- Attenuated Euclidean decoding:
$AED(x, y_i) = \sqrt{\sum_{j=1}^{n} |y_i^j| \, |x^j| (x^j - y_i^j)^2}$
- Loss-based decoding:
$LB(\rho, v_i) = \sum_{j=1}^{n} L(y_i^j . f^j(\rho))$,
Where ρ is a test sample, L is a loss function, and f is a real-valued function f: $R^n \rightarrow R$.
- Probabilistic-based decoding:
$PD(y_i, x) = -\log(\pi_j \in [1,\dots,n] : M_c(i, j) \neq 0$
$P(x^j = M_c(i, j) \mid f^j) + K)$,
Where K is a constant factor that collects the probability mass dispersed on the invalid codes, and the probability $P(x^j = Mc\ (i, j) \mid f^j)$ is estimated by means of

$$P(x^j = y_i^j \mid f^j) = 1/1 + e^{y_i^j (v^j f^j + w^j)} ,$$

where vectors υ and ω are obtained by solving an optimization problem (Passerini et al., 2004).

- Laplacian decoding:
$LAP(x, y_i) = \dfrac{\alpha_i + 1}{\alpha_i + \beta_i + K}$, where α_i is the number of matched positions between x and y_i, β_i is the number of miss-matches without considering the positions coded by 0, and K is an integer value that codifies the number of classes considered by the classifier.
- Pessimistic β-Density Distribution decoding:
Accuracy $s_i : \int_{vi-si}^{vi} \psi i(v, \alpha_i, \beta_i) dv = \frac{1}{3}$, where
$\psi_i(v, \alpha_i, \beta_i) = \frac{1}{K} v^{\alpha i}(1-v)^{\beta i}, \psi_i$ is the β-Density Distribution between a codeword x and a class codeword y_i for class c_i, and v $\in R : [0,1]$.
- Loss-Weighted decoding:
$LW(\rho, i) = \sum_{j=1}^{n} M_W(i, j) L(y_i^j . f(\rho, j))$,
Where
$M_w(i, j) = H(i, j) / \sum_{j=1}^{n} H(i, j)$,
$H(i, j) = 1/m_i \sum_{k=1}^{m_i} \varphi(h^j(\rho_k^i), i, j)$,
$$\varphi(x^j, i, j) = \begin{cases} -1, x^j = y_i^j \\ 0, otherwise \end{cases}$$
m_i is the number of training samples from class C_i, and ρ_k^i is the kth sample from class C_i.

Outline Of Ecoc Algorithm

Training

Load training data and parameters, i.e., the length of code L and training class K.

1. Create a L-bit code for the K classes using a kind of coding algorithm.
2. For each bit, train the base classifier using the binary class (0 and 1) over the total training data.

Testing

1. Apply each of the L classifiers to the test example.

2. Assign the test example the class with the largest votes.

What Makes A Good Ecoc?

The key problem for ECOC approach is how to design the coding matrix **M**. Many studies [10, 11, 12, 13, and 14] have shown that the final classifier will have good discriminate ability if the coding matrix **M** has the following characteristics:

- Characteristic 1: Row separation

Each codeword (a row in the coding matrix **M**) should be well-separated in Hamming distance from each of the other code words.

- Characteristic 2: Column separation

Each column should be uncorrelated with one another.
This means that the binary classifiers of different columns have low correlations among them.

- Characteristic 3: Binary classifiers have low Errors

While for recognition of a large number of classes, besides classification accuracy, the efficiency is also quite important. To make a quick decision, it is expected to evaluate as few binary classifiers as possible. This requires the codeword to be efficient (*i.e.* contains a small number of bits). As explained in [15], for a code to be efficient, different bits should be independent of each other, and each bit has a 50% chance of being one or zero. In ECOC design, independent bits can be relaxed as uncorrelated columns (*i.e.* property 2 mentioned above). And 50% chance of firing for each bit requires.

- Characteristic 4: Balanced column

For each column i, the numbers of 1 and -1 are equal,*i.e.*, $\sum_R M(r,i) = 0$.

Finding an ECOC satisfying the above characteristics is a NP-hard problem [16]. So we can say that for efficient and accurate recognition of a large number of classes, a good ECOC is expected to have the following characteristics:

- Efficient - requires a small number of bits.
- Good diversity - the coding matrix has good row and column separation.
- The resulting binary classifiers are accurate.

What's So Good About Ecoc?

1. Improves classification accuracy.
2. Can be used with many different classifiers.
3. Commonly used in many areas.
4. Not prone to over fitting.
5. Possibly try a variant.

Practical Advantages Of Ecoc

1. It is fast, simple and easy to program
2. It is flexible — can combine with any learning algorithm
3. Able to reduce the bias and variance produced by the learning algorithm. So it widely used to deal with multi-class categorization problems.
4. Low computational cost.
5. Outperforms the direct multiclass method.
6. Can use with data that is textual, numeric, discrete, etc.
7. General learning scheme - can be used for various learning
tasks.
8. Good generalization.

Disadvantages

1. ECOC is not effective if each individual codeword is not separated from each of the other code words with a large Hamming distance.
2. ECOC only succeed if the errors made in the individual bit positions are relatively uncorrelated, so that the numbers of simultaneous errors in many bit positions is small. If there are many simultaneous errors, the ECOC will not able to correct them (Peterson & Weldon, 1972).
3. ECOC support vector machines are not always superior to one-against-all fuzzy support vector machines.
4. One-versus-all schemes are more stable than other ECOC schemes.
5. Sometimes decomposition of multi-class problem into multiple binary problems we are doing in ECOC incurs considerable bias for centroid classifier, which results in noticeable degradation of performance for centroid classifier.
6. Finding the optimal ECOC is NP hard.

Comparison Of Some Ecoc Methods.

- One-Versus-All strategy.

The most well-known binary coding strategies are the one-versus-all strategy [17], where each class is discriminated against the rest of classes. In Fig. 1a, the one-versus-all ECOC design for a four-class problem is shown. The white regions of the coding matrix M correspond to the positions coded by 1 and the black regions to -1. Thus, the code word for class C_1 is {1,-1,-1,-1}. Each column i of the coding matrix codifies a binary problem learned by its corresponding dichotomizer h_i. For instance, dichotomizer h_1 learns C_1 against classes C_2, C_3, and C_4, dichotomizer h_2 learns C_2 against classes C_1, C_3, and C_4, etc.

- The Dense Random Strategy.

The dense random strategy [10], where a random matrix M is generated, maximizing the rows and columns separability in terms of the Hamming distance [7].

An example of a dense random matrix for a four class problem is shown in Fig. 1c.

- One-Versus-One and Random Sparse Strategy.

It was when Allwein et al. [10] introduced a third symbol (the zero symbol) in the coding process when the coding step received special attention. This symbol increases the number of partitions of classes to be considered in a ternary ECOC framework by allowing some classes to be ignored. Then, the ternary coding matrix becomes $M \in \{-1,0,1\}^{N\times n}$ .In this case, the symbol zero means that a particular class is not considered by a certain binary classifier. Thanks to this, strategies such as one-versus-one [20] and random sparse coding [10] can be formulated in the framework. Fig. 1b shows the one-versus-one ECOC configuration for a four-class problem. In this case, the gray positions correspond to the zero symbol. A possible sparse random matrix for a four-class problem is shown in Fig. 1d.

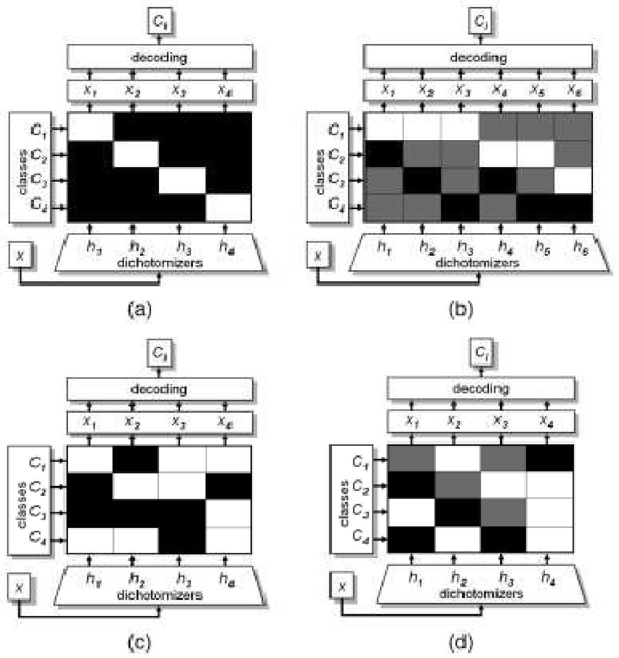


Fig. 1. (a) One-versus-all, (b) one-versus-one, (c) dense random, and (d) sparse random ECOC designs.

- *Spectral Error Correcting Output Codes* for Efficient Multiclass Recognition.

Algorithm:

Input: Given the class set C = $\{c_1, c_2, \ldots, c_n\}$

1. Train a SVM classifier f_{ij} for each class pair $\{c_i, c_j\}$
2. Construct the similarity graph G. Set each class c_i as a vertex and the weight w_{ij}.
3. Compute the normalized Laplacian $\mathbf{L}sym$ of G.
4. Compute the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of $\mathbf{L}sym$.
5. Transform each $\mathbf{v}_i$, $i \geq 2$, to a partition indicator vector $\mathbf{m}_i$
6. Generate an ECOC matrix $\mathbf{M}_l$ with code length l: $\mathbf{M}_l = [\mathbf{m}_{2,\ m3}, \ldots, \mathbf{ml}_{+1}]$
7. Train binary classifiers $\{fi\}_{i=1}^{l}$ to form code prediction function $\mathbf{f}_l(.) = [f_1(.), f_2(.), \ldots, f_l(.)]$
8. Search the optimal code length l^*.

Output: $\mathbf{M}_l^*$ and $\mathbf{f}_l^*(.)$

TABLE I
EFFICIENCY COMPARISON ON FACE RECOGNITION AND FLOWER CLASSIFICATION DATA.

Dataset	*Methods*	*Accuracy*	*Code length*
face	one-against-	99.5%	300

recognition with *k* = 18	all		
	Spectral ECOC	99.1%	30
	Random dense	98.9%	120
	Random sparse	98.8%	100
	Class Map	97.3%	150
	Discriminate ECOC	69.0%	200
flower classification with k=18	one-against-all	54.6%	102
	Spectral ECOC	54.7%	15
	Random dense	54.0%	83
	Random sparse	54.1%	44
	Class Map	50.2%	35
	Discriminate ECOC	35.2%	100

Conclusions

In this paper the different coding and decoding methods for Error Correcting Output Code have been studied. Advantages and disadvantages of some ECOC coding methods are discussed. From this study on ECOC one can conclude that compare to other methods, better performance can be achieved by using Error Correcting Output Code.

TABLE II
performance parameters (based on results obtained using weka on dataset contact-lenses with 10 folds cross validation)

Method Name	*Time Taken (Seconds)*	*Correctly Classified Instances (%)*	*Incorrectly Classified Instances (%)*	*Kappa Statistic*	*Mean Absolute Error*	*Root Mean Square Error*	*Relative Absolute Error (%)*	*Root relative squared error (%)*
Multiclass Classification using ECOC	0.02	75	25	0.5017	0.3263	0.3676	86.3777	84.1661

SVM(functions.SMO)	0.04	70.8333	29.1667	0.4381	0.3148	0.4082	83.3333	93.4754
ANN (functions. Multilayer Perceptron)	1	70.8333	29.1667	0.4766	0.2072	0.393	54.8401	89.9795
Meta Bagging	0.02	66.6667	33.3333	0.3356	0.269	0.3698	71.2054	84.6755
Meta AdaboostM1	0.01	70.8333	29.1667	0.5015	0.3581	0.4074	94.7807	93.2859
Nested Dichotomies Stacking	0	62.5	37.5	0	0.3778	0.4367	100	100

Reference

[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] 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.

[5] T. Hastie and R. Tibshirani, “Classification by Pairwise Coupling,” Annals of Statistics, vol. 26, no. 2, pp. 451- 471, 1998.

[6] 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.

[7] 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.

[8] R.E. Schapire, “Using Output Codes to Boost Multiclass Learning Problems,” Machine Learning: Proc. 14th Int’l Conf., pp. 313-321, 1997.

[9] 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.

[10] E. L. Allwein and R. E. Schapire. Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research, 1:113–141, 2000.

[11] N. Garc´ıa-Pedrajas and C. Fyfe. Evolving output codes for multiclass problems. IEEE Trans. Evolutionary Computation, 12(1):93–106, 2008.

[12] R. Ghaderi and T.Windeatt. Circular ecoc: A theoretical and experimental analysis. In ICPR, pages 2203–2206, 2000.

[13] O. Pujol and P. Radeva. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. PAMI, 28(6):1007–1012, 2006.

[14] R. Schapir and Y. Singer. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 2:263–286, 1995.

[15] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In NIPS, 2008.

[16] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. Machine Learning, 47(2-3):201–233, 2002.

[17] N.J. Nilsson, Learning Machines. McGraw-Hill, 1965.

[18] T. Hastie and R. Tibshirani, “Classification by Pairwise Grouping,” Proc. Neural Information Processing Systems Conf., vol. 26,pp. 451-471, 1998.