IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Survey Paper on Error-Correcting Output Code & Pessimistic β-Density Decoding Technique

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

Abstract

Error correcting output codes (ECOC) represent a successful extension of binary classifiers to address the multiclass problem. Lately, the ECOC framework was extended from the binary to the ternary case to allow classes to be ignored by a certain classifier, allowing in this way to increase the number of possible dichotomies to be selected. we show that by a special treatment procedure of zeros, and adjusting the weights at the rest of coded positions, the accuracy of the system can be increased. Besides, we extend the main state-of-art decoding strategies from the binary to the ternary case, and use two novel approaches: Laplacian and Pessimistic Beta Density Probability approaches. Our main research is to show that the ternary decoding techniques proposed outperform the standard decoding strategies. In this survey paper, we present an open source Error-Correcting Output Codes (ECOC) library. The ECOC framework is a powerful tool to deal with multiclass categorization problems. This library contains both 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) with the parameters defined by the authors, as well as the option to include your own coding, decoding, and base classifier.

Keywords: error-correcting output codes, multi-class classification, coding, decoding, open source, matlab, octave

ECOC

The basis of the ECOC framework is to create a codeword for each of the N_c classes. Arranging the codewords as rows of a matrix, we define a "coding matrix" *M*, where $M \in \{-1, 0, +1\}^{Nc\times n}$, in the ternary case, being n the code length. From point of view of learning, *M* is constructed by considering *n* binary problems (dichotomies), each corresponding to a matrix column. Joining classes in sets, each dichotomy defines a partition of classes (coded by +1, 0 or -1, according to their class set membership). In fig. 1 we show an example of a ternary matrix M. The matrix is coded using 7 dichotomies h1,…, h7 for a four multiclass problem (c_1, c_2, c_3, and c_4). The white regions are coded by 1 (considered as positive for its respective dichotomy, hi), the dark regions by -1 (considered as negative), and the grey regions correspond to the zero symbol (not considered classes for the current dichotomy). For example, the first classifier is trained to discriminate c_3 versus c_1 and c_2, the second one classifies c_2 versus c_1, c_3 and c_4, and so on. Applying the n trained binary classifiers, a code is obtained for each data point in the test set. This code is compared to the base codeword's of each class defined in the matrix M, and the data point is assigned to the class with the "closest" codeword.

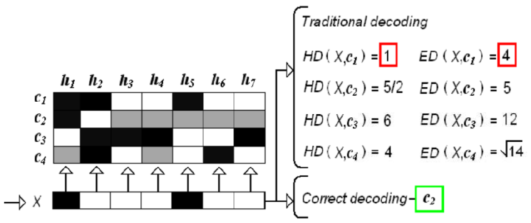

Figure. 1. Example of ternary matrix M for a 4-class problem. A new test codeword is misclassified due to the confusion of using the traditional decoding strategies.

The Error-Correcting Output Codes (ECOC) framework [9] is a simple but powerful framework to deal with the multi-class categorization problem based on the embedding of binary classifiers. Given a set of N_c classes, the basis of the ECOC framework consists of designing a codeword for each of the classes. These codeword's encode the membership information of each class for a given binary problem. Arranging the codeword's as rows of a matrix, we obtain a "coding matrix" Mc, where $Mc \in \{-1, 0, 1\}_{Nc\times n}$, being n the length of the codeword's codifying each class. 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. Each of these binary problems (or dichotomizers) splits the set of classes in two partitions (coded by +1 or -1 in Mc according to their class set membership, or 0 if the class is not considered by the current binary problem). Then, at the decoding step, applying the n trained binary classifiers, a code is obtained for each data point in the test set. This code is compared to the base codeword's of each class defined in the matrix Mc, and the data point is assigned to the class with the "closest" codeword. Several decoding strategies have been proposed in literature. The reader is referred to [7], [12] for a more detailed review. An example of an ECOC design is described in Fig. 2.

The ECOC designs are independent of the base classifier applied. They involve error-correcting properties [9] and have shown to be able to reduce the bias and variance produced by the learning algorithm [9]. Because of these reasons, ECOCs have been widely used to deal with multi-class categorization problems.

Library Algorithms

The ECOCs library is a Matlab/Octave code under the open source GPL license (gpl) with the implementation of the state-of-the-art coding and decoding ECOC designs. A main function defines the multi-class data, coding, decoding, and base classifier. A list of parameters is also included in order to tune the different strategies. In addition to the implemented coding and decoding designs, which are described in the following section, the user can include his own coding, decoding, and base classifier as defined in the user guide.

Implemented Coding Design

The ECOC designs of the ECOC library cover 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 in-formation of the particular domain is used to guide the coding design.

Problem- Independent ECOC Design

- One-versus-all [13]: N_c dichotomizers are learnt for N_c classes, where each one splits one class from the rest of classes.
- One-versus-one [10]: $n = N_c(N_c - 1)/2$ dichotomizers are learnt for N_c classes, splitting each possible pair of classes.
- Dense Random [3]: $n = 10 \cdot \log Nc$ dichotomizers are suggested to be learnt for Nc 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 rand om matrices, the one which maximizes a decoding measure among all possible rows of Mc is selected.
- Sparse Random [8]: n = 15 .log Nc dichotomizers are suggested to be learnt for Nc classes, where P(0) = 1 − P(−1) − P(+1), defining a set of random matrices Mc and selecting the one which maximizes a decoding measure among all possible rows of Mc.

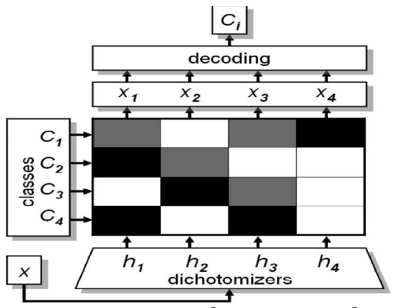

Figure 2: ECOC design example

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 $x = \{x1, ..,xn\}$ is compared with the class codeword's $\{C1, ..C4\}$, classifying the new sample by the class ci which codeword minimizes the decoding measure.

Problem – Dependent ECOC Design

- DECOC [11]: 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 [6]: problem-dependent design that uses $n = (N_c - 1) \cdot T$ di-chotomizers, 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 [12]: 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 c lasses.

Implemented Decoding Designs

The software comes with a complete set of ECOC decoding strategies. The notation used refers to that used in [7]:

- 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:

  $PD(y_i, x) = -\log(\pi_j \in [1,...,n] : M_c(i, j) \neq 0$

  $P(x^j = M_c(i, j) \mid f^j) + K)$,

  Where *K* is a constant factor that collects the probability mass dispersed on the invalid codes, and the probability $P(x^j = Mc\ (i, j) | f^j)$ 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 [20].
- 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

  $\psi_i(v, \alpha_i, \beta_i) = \frac{1}{K} v^{\alpha i}(1-v)^{\beta i}$, ψ_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

1. Load training data and parameters, i.e., the length of code L and training class K.
2. Create a L-bit code for the K classes using a kind of coding algorithm.
3. 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 [14, 15, 16, 17, and 18] 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 [19], 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 [20]. 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 [10], where each class is discriminated against the rest of classes. In Fig. a, 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 [14], where a random matrix M is generated, maximizing the rows and columns separability in terms of the Hamming distance [5].

- One-Versus-One and Random Sparse Strategy.

It was when Allwein et al. [14] 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 and random sparse coding [10] can be formulated in the framework. Fig. b 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.3 d.

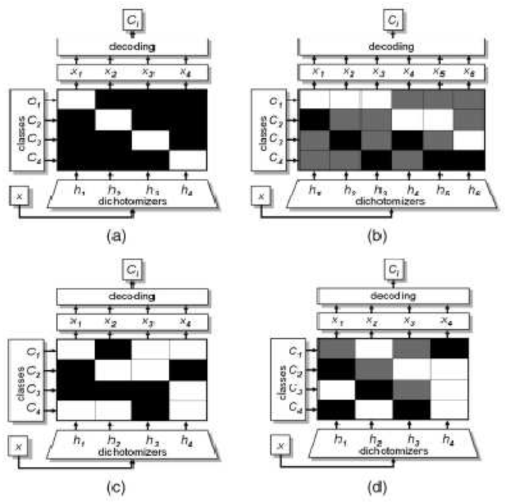

Figure 3(a) One-versus-all, (b) one-versus-one, (c) dense random, and (d) sparse random ECOC designs.

- *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 $\mathbf{f}_l(.) = [f_1(.), f_2(.), \ldots, f_l(.)]$
7. Search the optimal code length $l*$.

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

Implementation Details

The ECOCs Library comes with detailed documentation. A user guide describes the usage of the software. All the strategies and parameters used in the functions and files are described in detail. The user guide also presents examples of variable setting and execution, including a demo file. About the computational complexity, the training and testing time depends on the data size, coding and decoding algorithms, as well as the base classifier used in the ECOC design.

Conclusion

The ternary ECOC when applying a decoding strategy has not been previously enough analyzed. In this paper the different Ecoc coding and other decoding methods for Error Correcting Output Code have been studied. Advantages and disadvantages of some ECOC coding methods are discussed. We have seen two new decoding strategies that outperform the traditional decoding strategies when the percentage of zeros is increased. The validation of the decoding strategies at UCI repository databases gives an idea about the techniques that are more useful depending of the sparseness of the ECOC matrix M, where our proposed Pessimistic Density Probability and Laplacian strategies obtain the best ranking in the general case. From this study on ECOC one can conclude that compare to other methods, better performance can be achieved by using Error Correcting Output Code with above decoding methods.

References

[1] URL http://www.gnu.org/licences/.

[2] Sergio Escalera, Oriol Pujol, Petia Radeva : Error-Correcting Ouput Codes Library. Journal of Machine Learning Research 11 (2010) 661-664 .

[3] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2002.

[4] Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* , 2:263–282, 1995.

[5] Escalera, Oriol Pujol, and Petia Radeva. Boosted landmarks of contextual descriptors and Forest-ECOC: A novel framework to detect and classify objects in clutter scenes. *Pattern Recognition Letters*, 28(13):1759–1768, 2007.

[6] S. Escalera, O. Pujol, and P. Radeva. On the decoding process in ternary error-correcting output codes. *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 99, 2008.

[7] S. Escalera, O. Pujol, and P. Radeva. Separability of ternary codes for sparse designs of error-correcting output codes. *Pattern Recognition Letters*, 30:285–297, 2009.

[8] E. B. Kong and T. G. Dietterich. Error-correcting output coding corrects bias and variance. *Inter-national Conference of Machine Learning*, pages 313–321, 1995.

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

[10] O. Pujol, P. Radeva, , and J. Vitria`. Discriminant ECOC: A heuristic method for

application depen-dent design of error correcting output codes. *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 28:1001–1007, 2006.

[11] O. Pujol, S. Escalera, and P. Radeva. An incremental node embedding technique for error-correcting output codes. *Pattern Recognition*, 4:713–725, 2008.

[12] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[13] 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.

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

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

[16] 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

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

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

[19] A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. *IEEE Transactions on Neural Networks*, 15(1):45–54, 2004.

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