Journal of Machine Learning Research 11 (2010) 661-664 Submitted 8/09; Revised 1/10; Published 2/10

Error-Correcting Ouput Codes Library

Sergio Escalera SERGIO@MAIA.UB.ES
Oriol Pujol ORIOL@MAIA.UB.ES
Petia Radeva PETIA.IVANOVA@UB.EDU
Computer Vision Center
Edifici O, Campus UAB, 08193, Bellaterra, Barcelona, Spain

Editor: Cheng Soon Ong

Abstract

In this paper, we present an open source Error-Correcting Output Codes (ECOC) library. The ECOC framework is a powerful tool to deal with multi-class 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.

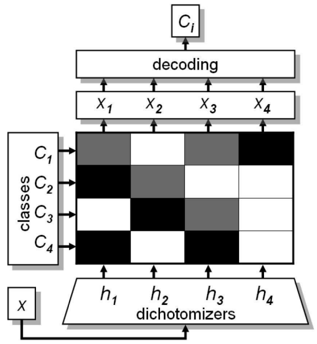
Keywords: error-correcting output codes, multi-class classification, coding, decoding, open source, matlab, octave

1. Error-Correcting Output Codes

The Error-Correcting Output Codes (ECOC) framework (Dietterich and Bakiri, 1995) 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 codewords encode the membership information of each class for a given binary problem. Arranging the codewords 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 codewords 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 M_c 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 codewords of each class defined in the matrix M_c, 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 Escalera et al. (2008) for a more detailed review. An example of an ECOC design is described in Fig. 1.

The ECOC designs are independent of the base classifier applied. They involve error-correcting properties (Dietterich and Bakiri, 1995) and have shown to be able to reduce the bias and variance produced by the learning algorithm (Kong and Dietterich, 1995). Because of these reasons, ECOCs have been widely used to deal with multi-class categorization problems.

©2010 Sergio Escalera, Oriol Pujol and Petia Radeva.



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 $x = \{x_1,..,x_n\}$ is compared with the class codewords $\{C_1,..C_4\}$, classifying the new sample by the class c_i which codeword minimizes the decoding measure.

Figure 1: ECOC design example.

2. 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 are 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.

2.1 Implemented Coding Designs

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 information of the particular domain is used to guide the coding design.

2.1.1 Problem-independent ECOC Designs

• One-versus-all (Rifkin and Klautau, 2004): N_c dichotomizers are learnt for N_c 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.

2.1.2 PROBLEM-DEPENDENT ECOC 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.

2.2 Implemented Decoding Designs

The software comes with a complete set of ECOC decoding strategies. The notation used refers to that used in (Escalera et al., 2008):

• Hamming decoding: $HD(x,y_i) = \sum_{j=1}^{n}(1-\text{sign}(x^j \cdot y_i^j))/2$, being x a test codeword and y_i a codeword from M_c corresponding to class C_i.

• Inverse Hamming decoding: $IHD(x,y_i) = \max(\Delta^{-1}D^T)$, where $\Delta(i_1,i_2) = HD(y_{i_1},y_{i_2})$, and D is the vector of Hamming decoding values of the test codeword x for each of the base codewords y_i.

• Euclidean decoding: $ED(x,y_i) = \sqrt{\sum_{j=1}^{n}(x^j - y_i^j)^2}$.

• Attenuated Euclidean decoding: $AED(x,y_i) = \sqrt{\sum_{j=1}^{n} \mid y_i^j \mid\mid x^j \mid (x^j - y_i^j)^2}$.

• Loss-based decoding: $LB(\rho,y_i) = \sum_{j=1}^{n} L(y_i^j \cdot f^j(\rho))$, where ρ is a test sample, L is a loss-function, and f is a real-valued function $f : \mathcal{R}^n \to \mathcal{R}$.

• Probabilistic-based decoding:
$PD(y_i,x)$=$-log\left(\prod_{j\in[1,..,n]:M_c(i,j)\neq 0} P(x^j = M_c(i,j)|f^j)+K\right)$, where K is a constant factor that collects the probability mass dispersed on the invalid codes, and the probability $P(x^j = M_c(i,j)|f^j)$ is estimated by means of $P(x^j = y_i^j|f^j) = \frac{1}{1+\mathbf{e}^{y_i^j(\upsilon^j f^j+\omega^j)}}$, where vectors υ and ω are obtained by solving an optimization problem (Passerini et al., 2004).

• Laplacian decoding: $LAP(x,y_i) = \frac{\alpha_i+1}{\alpha_i+\beta_i+K}$, 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 $s_i : \quad \int_{\nu_i - s_i}^{\nu_i} \psi_i(\nu,\alpha_i,\beta_i)d\nu = \frac{1}{3}$, where $\psi_i(\nu,\alpha_i,\beta_i) = \frac{1}{K}\nu^{\alpha_i}(1-\nu)^{\beta_i}$, ψ_i is the β-Density Distribution between a codeword x and a class codeword y_i for class c_i, and $\nu \in \mathcal{R} : [0,1]$.

• Loss-Weighted decoding: $LW(\rho,i) = \sum_{j=1}^{n} M_W(i,j)L(y_i^j \cdot f(\rho,j))$, where $M_W(i,j) = \frac{H(i,j)}{\sum_{j=1}^{n} H(i,j)}$, $H(i,j) = \frac{1}{m_i}\sum_{k=1}^{m_i}\varphi(h^j(\rho_k^i),i,j), \varphi(x^j,i,j) = \begin{cases} 1, & \text{if} \quad x^j = y_i^j, \\ 0, & \text{otherwise.} \end{cases}$, m_i is the number of training samples from class C_i, and ρ_k^i is the kth sample from class C_i.

3. 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.

Acknowledgments

This work has been supported in part by projects TIN2009-14404-C02 and CONSOLIDER-INGENIO CSD 2007-00018.

References

URL `http://www.gnu.org/licences/`.

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.

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

S. 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.

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.

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.

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

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

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.

O. Pujol, P. Radeva, , and J. Vitrià. Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 28:1001–1007, 2006.

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

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