A Fast Approach to Improve Classification Performance of ECOC Classification Systems

Paolo Simeone[1], David M.J. Tax[2], Robert P.W. Duin[2], and Francesco Tortorella[1]

[1] DAEIMI, Università degli Studi di Cassino
Via G. Di Biasio 43, 03043 Cassino (FR), Italy
{paolo.simeone,tortorella}@unicas.it
[2] Information and Communication Theory Group
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
{D.M.J.Tax,R.P.W.Duin}@tudelft.nl

Abstract. Error correcting output coding is a well known technique to decompose a multi-class classification problem into a group of two-class problems which can be faced by using a combination of binary classifiers. Each of them is trained on a different dichotomy of the classes. The way the set of classes is mapped on this set of dichotomies may essentially influence the obtained performance. In this paper we present a new tool, the k-NN lookup table to optimize this mapping in a fast way and a fast procedure to change the dichotomies in a proper way. Experiments on artificial and public data sets show that the proposed procedure may significantly improve the ECOC performance in multi-class problems.

Keywords: multiple classifiers systems, ECOC, k-NN .

1 Introduction

A common technique to solve an M class problem is to break the problem into dichotomies. Error Correcting Output Coding (ECOC) has arisen as one of the most eligible candidates to substitute monolithic classifier with multiple outputs by an ensemble of binary classifier. The reasons are mainly based on the stronger theoretical roots and the good understanding we have of some binary classifiers. The method, as originally conceived by Dietterich and Bakiri[1], consists of associating a binary string of length L to each of the M classes of the problem. This collection of strings is arranged into a $M \times L$ *coding matrix*. Each column in this matrix defines a binary classification problem. When a new sample is classified by the L dichotomizers, a new binary string is obtained, which has to be matched with the existing M binary class codes, using a suitable decoding technique. This approach has proved to provide a reliable probability estimation and a concurrent reduction of both bias and variance[2][3].

The original motivation for this method is based on the error correcting capabilities of the codes and a great effort has been devoted to improve the coding effectiveness. Allwein et al.[4] introduced a random technique for distributing the labels into the coding matrix and defined a weighted decoding technique to improve the performance in terms

N. da Vitora Lobo et al. (Eds.): SSPR&SPR 2008, LNCS 5342, pp. 459–468, 2008.

of general classification error. The random codes proved to perform not worse than the full code (the code containing all possible dichotomies). This leaded to the conclusion that it is not possible to define a coding strategy which can fit every problem.

Different codes applied to the same data set can yield different results as the classes may have an asymmetric influence on the problem. Some combinations of classes can better not be used for a dichotomy as they demand a too complicated base classifier. So changing the mapping of the classes on the set of dichotomies defined by the coding matrix may influence the performance of the base classifiers and thereby their combination.

Although Crammer et al.[5] demonstrated that designing the best output codes for a data set, given the set of base classifiers, is a NP-complete problem, approximate solutions can be practised. The proposal is to define a general framework to relate the data distribution in the space to the dichotomies and to use it to simplify the binary subproblems defined by each columns of the code. The class distributions in the feature space mainly determine the complexity of a classification problem. Some classes are probably closer than others as well as overlapping in some areas. ECOC provides a new labeling for these classes for each dichotomy, so the idea is to benefit of those which distribute binary labels in a proper way, such that a given binary classifier easily solves the new two class problem.

This target can be reached by permuting the rows of the code and thus changing the association class - binary string. An improvement on the error rate of the classification system is then expected by reducing the binary errors for each dichotomizer of the system. In any case, the idea is extremely powerful, because it will turn useful to analyze particular coding schemes for ECOC, like the LDPC codes[6] for example, which can take advantage of some error correcting techniques of general coding theory.

The optimization of the mapping of classes on the set of dichotomies can be prohibitively slow if it is based on retraining the base classifiers for different mappings or on training a multiple outputs classifier[7]. Moreover, it needs a separate validation set. We propose a much faster tool, the k-NN look-up table which simulates by k-NN classification results the use of more advanced base-classifiers. This table looks at the classes of the nearest k samples for each samples of the set and provides an evaluation of how easily a sample can be misclassified.

2 ECOC Framework

The ECOC approach consists of associating an L-length binary string (*codeword*) to each of the M class of the problem. This collection of strings is organized as an $M \times L$ *coding matrix*. Every column of the code defines a different binary subproblem that has to be solved by a dichotomizer as shown in Table 1. In the training phase, the new binary labels are fed to the binary classifiers. In the operating phase, the trained dichotomizers provide a string $\mathbf{o}(\mathbf{x})$ of L outputs for each sample which has to be classified.

Single errors on some of the L outputs not necessarily lead to the wrong final decision. Some of them can be recovered during the decoding phase performed on the outputs. Consider the Hamming Distance between two words of the code, i.e. the number of labels they differ from each other:

Table 1. Example of an exhaustive coding matrix for a 5 classes problem

classes	codewords														
	$\mathbf{f}_1$	$\mathbf{f}_2$	$\mathbf{f}_3$	$\mathbf{f}_4$	$\mathbf{f}_5$	$\mathbf{f}_6$	$\mathbf{f}_7$	$\mathbf{f}_8$	$\mathbf{f}_9$	$\mathbf{f}_{10}$	$\mathbf{f}_{11}$	$\mathbf{f}_{12}$	$\mathbf{f}_{13}$	$\mathbf{f}_{14}$	$\mathbf{f}_{15}$
ω_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ω_2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
ω_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
ω_4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
ω_5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

$$D_H(\mathbf{c}_i, \mathbf{c}_j) = \sum_{h=1}^{L} |c_{ih} - c_{jh}|. \tag{1}$$

A decoding technique is applied at the output of the dichotomizers and a sample is assigned to the k-th class if the Hamming distance between the output word $\mathbf{o}(\mathbf{x})$ and the k-th codeword is minimum:

$$\omega_k = \arg\min_h D_H(\mathbf{c}_h, \mathbf{o}). \tag{2}$$

The minimum Hamming distance (MHD) between two codewords of a defined code $d = \min_{i,j} D_H(\mathbf{c}_i, \mathbf{c}_j)$ is the most important parameter to take into account, because it tells how many binary errors can be corrected by the decoding rule. If the MHD is d, the code is able to correct $\lfloor (d-1)/2 \rfloor$ single errors from the base classifiers. A good coding matrix should therefore have a large MHD as possible between the rows of the code. A second demand, as observed by Dietterich himself, is to have also a good distance between the columns of the code. Similar columns produce high correlated classifiers that can easily provide the same errors on the same sample.

It is worth noting that MHD approach ignores the magnitude of prediction. Another approach based on computing the loss function for each base classifier, combining these measures into a total loss and minimizing the value has been successfully proposed in [4]. Notice that the choice of the decoding approach is not important for this paper and it has been analyzed in other papers [8] so we opted for the easiest criterion defined in eq. 2.

3 Reducing the Complexity of the Binary Problems

ECOC operates by binarization of multi class labels. Our aim is to estimate how good these binarizations/groupings of labels are in the feature space. It is easy to imagine that some dichotomies lead to simpler binary problems, separable by less complex classifiers, and some other to more complex ones that are hard to solve and quite complicated for a linear classifier.

It is worth noting that, once a random coding matrix C for a multi class problem is defined, the association between class label and binary codeword has an important role in the final performance of the system for the above given reason. The problem can be seen as a matter of finding the best grouping out of all possible labelings. The main

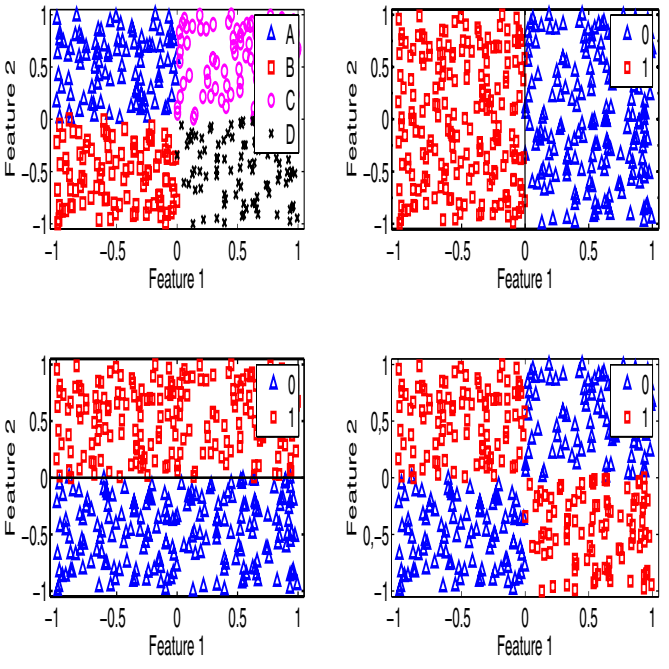


Fig. 1. An example of a 4 classes problem and some possible kind of binary coding for classes

idea is that a good relabeling is the one that groups together close classes in the feature space, so as to define the simplest possible binary problems.

In Fig. 1 a four class problem is shown in the top left corner. Each class is perfectly separable from the others. It is possible to solve this problem using the ECOC technique and given a low complexity classifier (a logistic one for example). Intuitively some grouping of labels are easier to solve. Putting together classes AB vs CD (top right), adopting a $[0011]^T$ dichotomy, or AC vs BD (bottom left), adopting a $[0101]^T$ dichotomy, the logistic classifier easily solves the problem, while it is not able to come up with a solution for the dichotomy $[1001]^T$ (bottom right).

What is needed is a tool to evaluate the difference between these dichotomies and possibly rank them. Notice that we assume that the coding matrix is given, and that we only want to optimize the association of the rows with the classes. We do not consider the problem of finding the best coding matrix as in some previous works[1][4].

Moreover it is useful to move toward an optimized view of the problem. If it is possible to define a tool to evaluate the re-labeling effects of the matrix, it will be possible to understand which dichotomies are good and which are not for a certain data distribution. The general problem of ECOC can be somewhat reformulated. It can be no more necessary to train a lot of classifiers to have very long codewords to be safe against binary errors, but, employing a suitable relabeling, few classifiers can grant the same performance obtained from a longer code. On the other hand simpler dichotomies imply the possibility to use simpler base classifiers. This is an additional reason to choose a linear classifier in the experiments.

Once fixed a code C for a M classes problem a fast way to change groupings of labels is to permute the rows to change the association multi class label - binary label. We look for the best of this association class-row by evaluating a measure to rank all the possible matrices obtained by permutations of the rows. A measure $\overline{\mathbf{m}}$ to distinguish the "quality" of the binary subproblems will be defined in the next section.

Moreover, it is worth noting that not all the possible permutations for a fixed coding matrix need to be examined, because a lot of them are effectively the same matrix with the same columns/dichotomies, but with different indexes. In order to simplify the search, we use a simple procedure of switching only two rows and analyze the resulting coding matrix. This means that, on the first stage of the algorithm, $M \times (M-1)$ possible codes are created by switching two rows and the top one in terms of the quality parameter $\overline{\mathbf{m}}$ is picked. The procedure which switches two rows is repeated on the new candidate code until there is no improvement. The following algorithm summarizes our procedure:

Algorithm 1. Permute and Search C

```
Compute m̄ for C
while m̄ is not improved do
  for i = 1 to M do
    for j = 1 to M, j ≠ i do
      Swap i-th and j-th rows generating C*
      Compute the new quality measure m̄*
      if m̄* < m̄ then
        Set C = C*, m̄ = m̄*
      end if
    end for
  end for
end while
```

4 Measuring the Quality of the Coding

The next step is to define how to measure the parameter $\overline{\mathbf{m}}$ for algorithm 1. To this aim, a *k-NN look-up table* (example shown on the left of fig. 2) is built, where, for each sample of the set, all the labels of the first k nearest neighbors are stored.

The defined coding matrix C gives the correspondence between the multi class labels and the binary ones for each dichotomy. Consider a training set $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i$ is a sample and $y_i \in \{1, .., M\}$ is the class label. As shown in fig. 2, for each dichotomy j of the coding matrix, the multi class label of the k-NN look-up table can be substituted by the binary one and the k-NN rule (f_j) can be evaluated for each sample. The error rate is then computed by:

$$\varepsilon_j = \frac{1}{N} \sum_{i=1}^{N} I(f_j(\mathbf{x}_i) \neq c_{y_i j}) \text{ for } j = 1, ..., L. \tag{3}$$

It is worth noting that this procedure does not go through massive training and combining of base classifiers, but the most expensive computations are done on the look-up

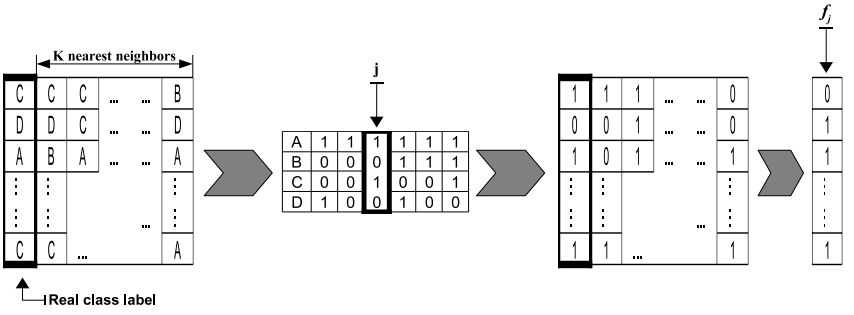


Fig. 2. k-NN look-up table

table itself. The value we wish to minimize is then the mean of this error rate over the L dichotomies:

$$\overline{\mathbf{m}} = \frac{1}{L}\sum_{j=1}^{L}\varepsilon_j. \tag{4}$$

It could be argued that also the error rate on computing the multi class labels could be computed at this stage. By evaluating the Hamming distance between the code and the collection of outputs of the binarized look-up tables a decoding technique has to be applied similarly to what is done with the ECOC itself. This measure has not been employed for two reasons. First the computation of the Hamming distance strongly enlarges the complexity of the framework, reducing all the benefits we had for the evaluation of many different codes in a fast way by replacing multi class labels with binary labels on the table. Second it is far too optimistic to think that this error rate computed at this stage and on the training set will behave like a typical ECOC scheme, which usually employs dichotomizers other than k-NN classifiers.

Attention must be paid to the choice of k. A big k can generate a lot of errors for the minority classes, while a little k can not be helpful to understand if some dichotomies have better performances. Experiments showed that a good rule of thumb, for now, is to choose a k next to the number of samples of the minority class. This way it should be avoided the possibility that some classes are encapsulated by others, while the effectiveness of the method is still maintained.

5 Experiments

In order to validate our tool, experiments have been made for an artificial data set and for two different multi class data sets from the UCI repository [9]. Notice that the given search algorithm outputs a temporary best code for every $while$ iteration which evaluates $M \times (M-1)$ different matrices per time. Every output code for each of these iterations has been used as coding matrix for an ECOC system. We employed a logistic linear classifier as base classifier embedding the ECOC framework in the PR tools suite (`http://www.prtools.org`). Then, we observed the evolution of the test error at each step and studied how effective our tool is in providing a reliable evaluation for

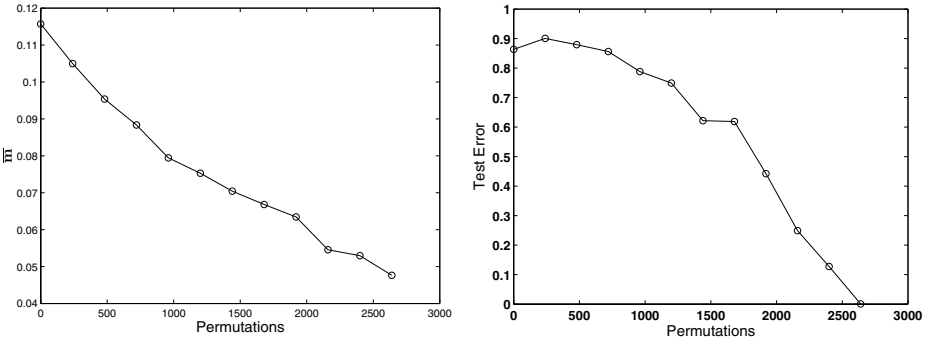
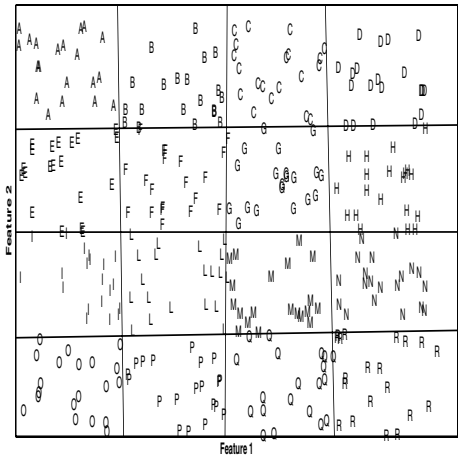


Fig. 3. Concurrent reduction of $\overline{\mathbf{m}}$ and of the test error for the artificial dataset

coding and dichotomies. In fig. 3 an example of the concurrent reduction of the quality parameter $\overline{\mathbf{m}}$ and of the test error for each $while$ iteration of the algorithm 1 is shown for one run of experiments on the artificial data set. On the rest of the experimental section the results of training and test error are shown only at the starting and at the ending point of this algorithm.

5.1 Artificial Data Set

The artificial data set is formed by sixteen different classes uniformly distributed in a square region of a two dimensional feature space, which can be divided in a 4×4 pattern of squares as in fig. 4 (left). The ideal coding for this data set is represented by columns c_1 to c_6 of the code in fig. 4 (right), where c_1 to c_3 lead to three horizontal classifier and c_4 to c_6 to the three vertical, always shown on the same figure.



classes	$\mathbf{c_1}$	$\mathbf{c_2}$	$\mathbf{c_3}$	$\mathbf{c_4}$	$\mathbf{c_5}$	$\mathbf{c_6}$	$\mathbf{r_1}$	$\mathbf{r_2}$
A	1	1	1	0	0	0	1	1
B	1	1	1	1	0	0	1	1
C	1	1	1	1	1	0	1	1
D	1	1	1	1	1	1	1	1
E	0	1	1	0	0	0	0	1
F	0	1	1	1	0	0	1	1
G	0	1	1	1	1	0	1	1
H	0	1	1	1	1	1	1	0
I	0	0	1	0	0	0	0	1
L	0	0	1	1	0	0	0	1
M	0	0	1	1	1	0	1	0
N	0	0	1	1	1	1	1	0
O	0	0	0	0	0	0	0	1
P	0	0	0	1	0	0	0	0
Q	0	0	0	1	1	0	0	0
R	0	0	0	1	1	1	1	0

Fig. 4. Artificial data set in the two features space and a possible binary coding of the 16 classes

Table 2. Training and Test error for the Artificial Data Set: (a) ideal code (b) redundant code

(a)

Artificial Data set			
Training Error		Test Error	
0.7289	0.5045	0.7239	0.5042
0.9629	0.6869	0.9589	0.6815
0.8253	0.5453	0.8240	0.5546
0.9008	0.4865	0.9002	0.4998
0.8652	**0.0000**	0.8634	**0.0007**
0.9038	0.7737	0.9055	0.7719
0.9049	0.6224	0.9058	0.6254
0.8818	0.7569	0.8811	0.7583
0.9283	**0.0000**	0.9269	**0.0013**
0.8467	0.6858	0.8484	0.6875

(b)

Artificial Data set			
Training Error		Test Error	
0.8278	0.5298	0.8267	0.5289
0.9109	0.7355	0.9098	0.7339
0.8124	0.5449	0.8133	0.5519
0.9004	0.5656	0.8998	0.5436
0.8654	0.5553	0.8638	0.5656
0.8600	0.6303	0.8628	0.6384
0.8176	**0.3927**	0.8123	**0.3863**
0.8424	0.4178	0.8457	0.4030
0.8738	0.4381	0.8741	0.4310
0.8467	0.4637	0.8484	0.4711

We scramble ten times these multi class labels in order to have a casual row associated to a class and for each of these random initialization of the labels we run the search algorithm. We then wish to process and analyze different codings by permuting rows of these matrixes through the algorithm 1. Every time the optimum will be to reach the association multi class labels - binary labels shown in the figure, which correspond to an error rate of 0.0017.

Results in terms of error rate on the training and the test set are shown on table 2-(a): each row is one of the ten random initialization, while the train and test error column presents the values of the error rate, before and after running the algorithm 1. Other experiments were done with a redundant code, where two other columns are added to the ideal coding and corresponding to a diagonal split of the feature space. This coding is accomplished by the dichotomies r_1 and r_2 of the code in fig. 4 (right). Results of these experiments are on table 2-(b). The best error rate attainable with the entire code (c_1 - c_6 and r_1 - r_2) is 0.0785.

Observing the tables it is clear how the procedure always performs a reduction of both the error rates and we obtain the best performance on two runs for the code $c_1 - c_6$. In order to obtain the best association class-row we suggest then to carry out the random initialization procedure. The redundant code results also shows how the method always reduce the error rate, but the best performance is not reached because of the algorithm, which is only suboptimal and does not analyze a sufficient number of permutations to succeed. In both tables the best error rate attainable is bolded.

5.2 Public Data Sets

Differently from the previous set of experiments two real data sets from a public repository are chosen (characteristics showed in tab. 3) and ten cross validation folders are created for them. To fix the perfect choice of a coding matrix for the ECOC is beyond the purposes of this paper. We are interested in demonstrating that the tool is useful in understanding the best coding scheme for the ECOC. A random and short matrix

Table 3. Public data sets and coding matrices used in the experiments

Data Sets	# Classes	# Features	Coding Matrix	Length (L)	# Samples
SatImage	6	36	Random	15	5470
Letter	26	16	Die. 26-15	15	2970

is fixed for the SatImage data set, while for the Letter data set a code is selected from the set provided by Dietterich on his web page, just to make even more clear that the performance improvement does not involve the choice of the coding matrix. Even in this case, a multiple initialization procedure is done, randomly associating the classes to the rows of the coding, providing this way a different starting point for each run of the experiments.

Each row of table 4 is one of the initialization of the multi class labels, while train and test error columns are split into two columns: the first indicates the original error rate and the second is the error rate reached by the suboptimal procedure operated by the algorithm 1. It is clear that on SatImage data set (table 3-(a)) the error rate is always reduced as it happens for the parameter $\overline{\mathbf{m}}$, while this is not always true for the Letter data set (table 3-(b)).

These experiments show how the procedure is only suboptimal, because it always analyzes a number of coding matrices equal to $M \times (M-1) \times W$, where W is the number of $while$ cycles of the algorithm 1. For the Letter data set , for example, the total number of possible permutation of rows is 26!, while the algorithm tests $26 \times 25 \times 17$ different codes, i.e. a limited number of all the possible configurations. SatImage data set instead was analyzed for a number of combinations which was over the half of the number of permutations (6!) and the probability to find the best combination of rows is certainly higher. Another problem is in the choice of the base classifier, because it is probably hard for the logistic classifier to solve the binary problems defined by the dichotomies. This explains how the reduction of the quality parameter could not be in accordance with the reduction of the error rate sometimes. In both tables the best error rate attainable is bolded.

Table 4. Training and Test error for the for the Public Data Sets:(a) SatImage and (b) Letter

(a)

Satimage Data set			
Training Error		Test Error	
0.1956	**0.1710**	0.2041	**0.1786**
0.1925	**0.1710**	0.1960	**0.1786**
0.1962	0.1910	0.2014	0.1946
0.2005	0.1910	0.2053	0.1946
0.2230	**0.1710**	0.2306	**0.1786**
0.2003	0.1910	0.2057	0.1946
0.1725	**0.1710**	0.1805	**0.1786**
0.2018	**0.1710**	0.2047	**0.1786**
0.1889	**0.1710**	0.1940	**0.1786**
0.1926	0.1910	0.2005	0.1946

(b)

Letter Data set			
Training Error		Test Error	
0.6079	0.6188	0.6188	0.6289
0.6119	0.6124	0.6281	0.6239
0.6350	0.6150	0.6487	0.6217
0.6499	0.6285	0.6595	0.6333
0.6214	0.6235	0.6289	0.6373
0.6310	**0.5985**	0.6441	**0.6135**
0.6635	0.6302	0.6675	0.6439
0.6189	0.6352	0.6319	0.6492
0.6689	0.6284	0.6738	0.6412
0.6636	0.6117	0.6789	0.6304

6 Conclusions and Future Works

In this paper we proposed a fast and flexible tool to optimize the mapping between classes and the dichotomies defined in a chosen coding matrix. Both artificial and public data sets experiments validated our hypotheses. Using the random initialization procedure, on many cases we experienced the best performance. This suggests that many random initializations can be advantageous in order to find the best settings. A more appropriate and smart algorithm will be employed in the future works in order to fix the problems experimented with a huge data set like Letter.

Another known problem to fix is how to use the tool for an unbalanced data set. Future work will focus also on finding the optimal value of k in order to solve it. Moreover we will verify if the tool can be used to build a proper coding matrix from the data starting from the k-NN look-up table.

References

1. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263–286 (1995)
2. Kong, E.B., Dietterich, T.G.: Error-Correcting Output Coding Corrects Bias and Variance. In: International Conference on Machine Learning, pp. 313–321 (1995)
3. Kong, E.B., Dietterich, T.G.: Probability estimation via error-correcting output coding. In: Int. Conf. of Articial Inteligence and soft computing, Ban, Canada (1997)
4. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 9–16 (2000)
5. Crammer, K., Singer, Y.: On the Learnability and Design of Output Codes for Multiclass Problems. Machine Learning 47 (2002)
6. Marrocco, C., Simeone, P., Tortorella, F.: Embedding Reject Option in ECOC Through LDPC Codes. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 333–343. Springer, Heidelberg (2007)
7. Pujol, O., Radeva, P., Vitrià, J.: Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes. IEEE Trans. On Pattern Analysis And Machine Intelligence 28(6), 1001–1007 (2006)
8. Windeatt, T., Ghaderi, R.: Coding and decoding strategies for multi-class learning problems. Information Fusion 4(1), 11–21 (2003)
9. Blake, C., Keogh, E., Merz, C.J.: UCI Repository of Machine Learning Databases (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`