On the design of error adaptive ECOC algorithms

Elizabeth Tapia[1], José Carlos González[1], Javier García-Villalba[2]

[1]Department of Telematics Engineering - Technical University of Madrid, Spain
{etapia, jcg}@gsi.dit.upm.es
[2]Department of Computer Systems - Complutense University of Madrid (U.C.M.), Spain
javiergv@sip.ucm.es

Abstract. A new type of adaptive boosting algorithms suitable for multiclassification problems is proposed. By means of recursive coding models and the ECOC framework, learning in multivalued output domains is formulated in terms of binary learning algorithms. In such a system, learning algorithms play a decoding function. The use of recursive coding allows the design of error adaptive ECOC learning algorithms. This approach generalizes the boosting concept in arbitrary multivalued output domains in a transparent way and gives rise to the so-called RECOC learning models. A RECOC instance based on Turbo Codes is presented.

1 Introduction

Redundancy is a well-known concept in coding theory. Transmission of information through noisy channels can be carried out at arbitrarily small error rates by suitable channel encoding. Redundancy is also present in the learning field. It is observable in the repetition of concepts implemented by boosting [1] algorithms and it is explicitly used in Dietterich's Error Correcting Output Codes (ECOC) approach for classification problems in multivalued output domains [2]. In his work, Dietterich recognized that ECOC schemes should resemble random coding in order to achieve successful learning. Let us denote by M the number of classes involved. Good ECOC schemes for low to medium M values have been found by exhaustive search. However, the design of suitable ECOC schemes in arbitrary output domains has remained as an open problem. Despite of their common motivation, namely the design of low complexity learning algorithms, both ECOC and boosting appear to be conceptually different for Machine Learning literature. Boosting algorithms are assumed to exhibit an error adaptation capacity, which appears to be absent in ECOC algorithms. Because of this apparent lack of error adaptation capacity, ECOC algorithms have been used in the core of multiclass boosting algorithms. On the opposite side, binary boosting has not been used in the core of ECOC algorithms. This is a striking question considering that the ECOC strategy reduces M classification problems to a related set of binary ones.

In this paper, we address the design of *general error adaptive* ECOC schemes in arbitrary multivalued output domains. We do so by means of Recursive Error Correcting codes [3] under the ECOC framework. Resulting learning algorithms are inherently error adaptive as follows from the iterative decoding strategies required for recursive codes. Furthermore, learning under this new framework can be improved with inner binary boosting algorithms, thus providing a pleasant symmetry between binary boosting and ECOC algorithms.

The remainder of this paper is organized as follows. In section 2, we revisit the standard ECOC approach. In section 3, we present the so-called RECOC learning model as an extension of the ECOC approach with Recursive Error Correcting Codes. In section 4, we develop a particular instance of the RECOC learning based on turbo codes [4]. In section 5, experimental results are presented. Finally, in Section 6 conclusions and further work are presented.

2 ECOC Learning

In its standard form, the ECOC algorithm is applicable to the supervised learning of target classes $C : X \rightarrow Y$, $|Y| = M$, i.e. classification problems in M valued output domains. Each of the involved classes is assigned a binary codeword $\mathbf{c} = (c_o \quad ..., \quad ..., \quad c_{n-1})$ belonging to some specified coding scheme Θ i.e. a kind of source and channel-encoding mapping $\mathrm{E} : \mathrm{Y} \rightarrow \Theta$ is defined. Let S, $|S| = m$, be a training sample for some target concept $f \in C$. Having stated the E mapping, the ECOC approach constructs a vector a binary training samples $\mathbf{S} = [S_i]$, $1 \le i \le n$, by sequential application of the E mapping to each instance in S as shown in **Table 1**.

Table 1. The ECOC approach

ECOC							
Feature	Label	c_0	c_2		c_i		c_{n-1}
$\mathbf{x}_1$	y_1	0	1		0		1
$\mathbf{x}_2$	y_2	1	0		1		1
					0		
$\mathbf{x}_m$	y_m	1	1		1		0

Afterwards, an ensemble of n binary classifiers is constructed by subsequent training of a fixed binary learning algorithm WL with binary training samples S_i, $1 \le i \le n$. The classification of a new feature vector $\mathbf{x} \in X$ requires the evaluation of the set of n binary classifiers so that a "received" noisy codeword $\mathbf{h}(\mathbf{x}) = (h_0(\mathbf{x}), \ ..., \ h_{n-1}(\mathbf{x}))$ can be computed given that a target concept $\mathbf{c}(\mathbf{x}) = (c_0(\mathbf{x}), \ ..., \ c_{n-1}(\mathbf{x}))$ has been transmitted. The final class value assigned to the feature vector $\mathbf{x}$ is essentially the one pointed out by a Minimum Distance

(MD) decoder passed through an E^{-1} mapping. Therefore, the ECOC prediction rule can be stated as

$$h_f(\mathbf{x}) = \mathrm{E}^{-1}[\mathrm{MD}\ (\Theta, \mathbf{h}(\mathbf{x}))] \quad (1)$$

The aim of coding redundancy is to cope with the errors arising from the underlying binary learner algorithm. In [2], Dietterich established that the main consideration in the selection of Θ-ECOC schemes should be their ability to induce both random columns and random rows in the binary matrix shown in **Table 1**. In order to satisfy this need, he proposed the design of ECOC schemes based on Hamming distances between rows and columns. Good ECOC schemes under Hamming distance criteria were found by exhaustive search. However, the design of a general method for constructing good ECOC schemes valid for arbitrary M valued output domains has remained as open problem.

ECOC results are surprisingly good and they can be improved with higher block-coding lengths. It should be noted, however, that higher block-coding lengths are clearly an odd requirement for scalability purposes. In fact, this is a well-known problem in coding theory, where the trade off between block-coding length and error rates has been largely dealt with. Coding theory has been able to find a promising solution for these problems under the theory of Recursive Error Correcting Codes for which turbo codes are particular instances.

3 Recursive ECOC Learning

In the ECOC framework, the prediction stage involves essentially a Minimum Hamming distance decoding process. It should be noted, however, that this decoding approach ignores both the coding structure itself and the observed binary learners' reliability. In this way, ECOC decisions suffer from the profitable error adaptation capacity in learning algorithms.

Now, think about the learning of a target concept $f \in C: X \to Y$, $|Y| = M$, under the ECOC framework. Clearly, in order to transmit M output values, at least $k \geq \log_2 M$ bits are required. Therefore, an M valued target concept $f(x)$ can be broken down into k binary concepts by means of a suitable *quantization* Q_M process. Reliable transmission of these $k-$blocks of information bits through a noisy channel (induced by binary learners' drawbacks) can be achieved by suitable channel encoding.

Let us assume $M = 16$ and a simple $(8,4,4)$ [1] extended Hamming Θ coding scheme. The code is systematic[2] and has sixteen different codewords. Instead of assigning the ECOC codewords in a random way, let us consider an encoding stage

[1] Using standard coding notation, it denotes a (n, k, d) linear block code, being n the codeword length, k the number of information bits and d the minimum code distance

[2] First k bits are information bits

for each bitstream $\mathbf{s} = (u_0, u_1, u_2, u_3)$ representing a label in the M valued output domain Y. Each transmitted codeword $\mathbf{c} \in \Theta$ is of the form $\mathbf{c} = (u_0, u_1, u_2, u_3, z_4, z_5, z_6, z_7)$, with parity bits defined as follows[3]

$$\begin{aligned} z_4 + u_0 + u_1 + u_2 &= 0 \\ z_5 + u_0 + u_1 + u_3 &= 0 \\ z_5 + u_0 + u_2 + u_3 &= 0 \\ z_5 + u_1 + u_2 + u_3 &= 0 \end{aligned} \qquad \textbf{(2)}$$

Now, consider an alternative view of the prediction stage by the introduction of a recursive construction view for our simple $(8,4,4)$ extended Hamming code. Any codeword $\mathbf{c} = (u_0, u_1, u_2, u_3, z_4, z_5, z_6, z_7) \in \Theta$ can be thought as the 8-tuples, which simultaneously verifies the set of four parity check equations given in **(2)**. Let us develop a graphical model for the above set of parity equations defining the ECOC encoding process. Clearly, each equation in **(2)** can be considered as a simple parity check codes for the transmission of blocks of *four* information bits. Therefore, our $(8,4,4)$ code can also be defined in terms of four parity check subcodes. This recursive view of a code construction, i.e. a code built from component subcodes, can be modeled by means of the so-called Tanner graphs. The set of parity coding constraints S_t, $1 \le t \le 4$, and the codeword bits define the nodes in a graph while edges are put connecting constraint to variables (codeword bits). The Tanner graph for our (8,4,4) code is shown in **Fig. 1**

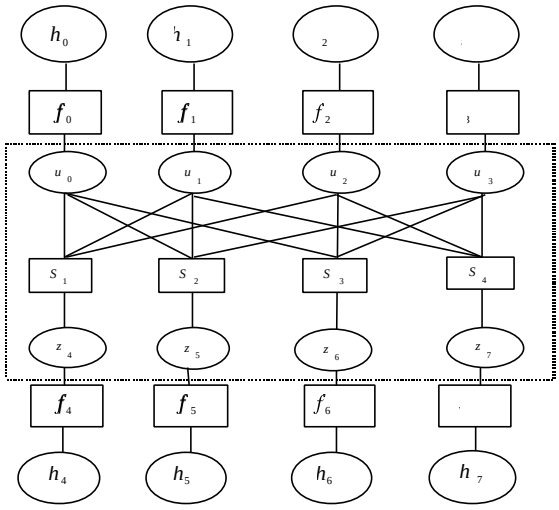

Fig. 1. Factor Graph for the (8,4,4) Hamming code

[3] All operations in the Galois Field GF(2)

A received noisy codeword $\mathbf{h} = (h_0,\ h_1,\ h_2,\ h_3,\ h_4,\ h_5,\ h_6,\ h_7)$ can be thought as the result of a random toggling process on the transmitted codeword bits in $\mathbf{c} = [c_i],\ 0 \le i \le 7$. A simple model for this toggling process is a binary discrete additive one.

$$h_i = c_i + e_i \quad 0 \le i \le 7 \tag{3}$$

each e_i being a binary random variable with $P(e_i = 1) = p_i$. In coding terms, this model of errors is no more the so-called *discrete memoryless channel*. In ECOC learning terms, it is equivalent to assuming that binary learner's errors are independent between each other. In this way,

$$P(h_i \neq c_i \mid c_i) = p_i \qquad 0 \le i \le 7 \tag{4}$$

In practical terms, we would demand learner's errors to be highly uncorrelated. The transmission of codewords through a noisy channel introduces further constraints on codewords bits. These constraints are those that model the toggling process. These new set of constraints can be directly introduced in a Tanner graph noting that we can define a function $f_i(c_i)$ such that

$$f_i(c_i) = P(h_i \mid c_i) = \begin{cases} p_i & h_i \neq c_i \\ 1 - p_i & h_i = c_i \end{cases} \qquad 0 \le i \le 7 \tag{5}$$

It should be noted that the definition of $f_i(c_i)$ assumes h_i as a parameter. This is because h_i variables have been instantiated at codeword reception time (they are the binary learners' predictions). In addition, information bits (binary target concepts defining the M valued one) can be constrained by a prior distribution $p(u_i)$, $0 \le i \le 3$. In the simplest case, this prior distribution can be assumed uniform so that $p(u_i = 1) = 0.5$ holds, $0 \le i \le 3$. The resulting extended Tanner graph is known as the code Factor graph (**Fig. 1**). Now, we are almost ready to formulate an error adaptive version of the ECOC algorithm. The objective is to compute probabilities $p(u_i \mid \Theta, f_o, .., f_{n-1}, p_o, ... p_{k-1})$, so that we can give an estimate u_i* of a transmitted binary concept u_i

$$u_i^* = \arg \max_{u_i} \left[p(u_i \mid \Theta, f_0, .., f_{n-1}, p_0, ... p_{k-1}) \right] \quad i = 0, ..., k-1 \tag{6}$$

If this were possible then a prediction in the M valued output domain can be taken directly from the estimated source vector $\mathbf{s}^* = (u_0 *,\ ....,\ u_{k-1} *)$ by application of the inverse quantization process Q_M^{-1}. The code factor graph shown in **Fig. 1** resembles too much a Bayesian network. In fact, Bayesian networks are a special case of factor graphs. Furthermore, computation of probabilities $p(u_i \mid \Theta, f_0, .., f_{n-1}, p_0, ... p_{k-1})$ can be done by a message-passing scheme similar

to the well-known Pearl's Belief propagation algorithm [5]. The observed results in the coding field are almost optimal, despite of the presence of cycles. Therefore, if we can devise a recursive coding view in the standard ECOC algorithm, then we can also give an error adaptive ECOC decision. This adaptive ECOC decision is constructed by a message-passing scheme in the underlying factor graph. The only requirement would be the knowledge of probabilities p_i, $0 \le i \le n-1$, governing the learning noise. However, these probabilities can be directly estimated from the training error responses achieved by the set of binary learners. From the ECOC point of view, each decoding process at a local component subcode issues a set of predictions on a subset of target information concepts. In this way, each binary target concept is observed and estimated in more than one context. Hopeful, these estimations can be combined. If that the case, the prediction stage may be solved for a binary distributed learning system. These considerations allow us to design *general error adaptive* Recursive ECOC (RECOC) algorithms. The *error adaptation* property follows directly from the use of training errors in computation of binary target predictions. The term *general* refers to the application of the RECOC model in arbitrary output domains without exhaustive search of ECOC schemes. This follows from the intrinsic pseudorandom behavior [6] of recursive codes. In the next section, an instance of the RECOC learning model based on Turbo Codes will be analyzed.

4 RECOC _Turbo Learning

Let us consider an instance of the RECOC model based on Turbo Codes. Each block of k bits representing an output label will be turbo encoded. Let us briefly explain the parameters defining a turbo-encoding scheme.

Definition 1 (Parallel Concatenated Code and Turbo Codes) *A Parallel Concatenated Code is formed by two or more constituent encoders joined by an interleaver. The inputs information bits feed the first encoder and then after having been scrambled with corresponding interleavers, enter another encoders. The codeword of a Parallel Concatenated Code consists of the input bits to the first encoder followed by the parity checks bits of other encoders. In most applications, Parallel Concatenated coding involves a number F of Recursive Systematic Convolutional Codes (RSCC) as constituent subcodes, each with memory B. The resulting codes have been named Turbo codes.*

A nice and clear introduction to RSCCs can also be found in [6]. In nearly all applications involving turbo codes, rate $r = \frac{1}{2}$ (one-information bit transmitted with a parity bit) RSCC component subcodes with small memory B are used. A typical value for the number of component subcodes is $F = 2$. In this case, a $r = \frac{1}{3}$ turbo code is obtained. The presence of block interleavers (denoted by P in **Fig. 2)** reduces the probability of low-weight turbo codewords improving the quality of the resulting code, even with simple component subcodes.

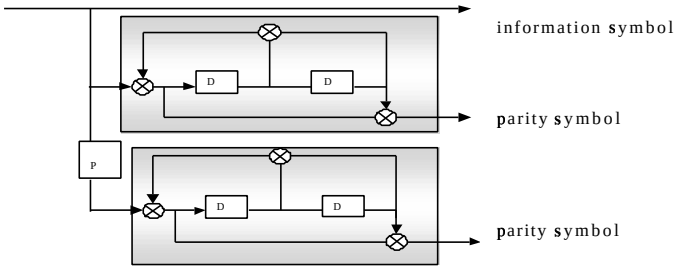

Fig. 2. $r = \frac{1}{3}$ Turbo Code from $F = 2$ component RSCCs with $B = 2$ units or memory

Because of the feedback structure in the RSCCs and the permutation of block of input symbols, the resulting code resembles a random coding scheme. This distinctive property makes them suitable for using in the ECOC framework. From a decoding point of view, a turbo coder behaves like a block code with codeword length $n = (F+1)\cdot(k+B)$ and rate $r = \frac{k}{n}$, being k the interleaver length and B de degree of memory present in each component RSCC. The main reason for Turbo Codes successful implementation is the availability of efficient algorithm for *iterative decoding* namely a recursive version of the BCJR algorithm [7] [8]. For the sake of brevity, we suggest the interested reader a review of the material available at http://www.ima.umn.edu/csg/. For purposes of this paper, once we have defined the required parameters, turbo decoding may be treated as a black box. Now, let us consider a RECOC instance by means of turbo codes. We will call this instance as the RECOC_Turbo learning algorithm.

RECOC_Turbo Algorithm

Input:

Quantizer Q_M with k bits per input in $\{1,...,M\}$

Turbo Code TC based on F components RSCC with memory B

Interleaver with length $k = \lceil \log_2 M \rceil$

Training Sample S with $|S| = m_S$

Distribution D on S, Binary Learner WL

Number of iterations I for Turbo Decoding

Processing:

$n = (F+1)\cdot(k+B)$, TurboECOC (S, Q_M, TC, $WL_0,.....,WL_{n-1}$, $e_0,...,e_{n-1}$)

Output:

$h_f(\mathbf{x}) = Q_M^{-1}(\text{Turbo\_Decoding}(\mathbf{TC}, \mathrm{I}, \mathbf{x}, WL_0,....,WL_{n-1}, e_0,...,e_{n-1}))$

End

The number of binary learners used in the binary ECOC expansion is set to the turbo codeword length, namely $n = (F+1)\cdot(k+B)$. The `TurboECOC` routine receives the training set S, the quantizer function Q_M and the turbo code structure **TC**. Consequently, it outputs a set of learners WL_i with their corresponding training errors in a vector $E = [e_i]$, $0 \le i \le n-1$. Let $h_i(\mathbf{x})$ be the prediction issued by the binary weak learner WL_i trained with the sample S_i under a probability distribution D_i, $0 \le i \le n-1$.

$$e_i = P_{S_i \approx D_i}\left[h_i(\mathbf{x}) \neq y\right] \quad (7)$$

The application of the `Turbo_Decoding` algorithm through **I** iteration steps throws an estimation of the systematic part of the transmitted codeword (first k binary concepts). The decoded information symbols are then passed to the inverse quantization process Q_M^{-1}, which then issues the final class value.

5 Experimental Results

We have tested five learning domains from the UCI repository database, namely Audiology, Anneal, Primary Tumor, Glass and Lymph. In all cases evaluation corresponds to a 66 % split of the original dataset. Learning algorithms implementations were developed using public domain Java WEKA [9] library. Therefore, AdaBoost (AB), Decision Stump[4] (DS) and C4.5[5] (denoted by C4.5b for the binary case) algorithms implementations details can be fully determined from WEKA documentation. For rate $r = \frac{1}{3}, \frac{1}{4}$ turbo coding and decoding, we developed WEKA library extension based on [8][11]. First experimental results pointed out that RECOC_Turbo performance depends strongly on the correct estimation of the error vector E. The first fact suggested to us that further improvements might be obtained by refining the predicting capabilities of binary learners i.e. by boosting them before performing the turbo-decoding algorithm. This conjecture was confirmed by experimental results. In most cases, a reduced number of inner binary boosting steps using the standard AdaBoost algorithm were sufficient to perform RECOC_Turbo learning with test error below C4.5 decision tree algorithm which was used as benchmark (similarly to [2]). In order to study the effect of memory, we first simulated three different turbo encoding schemes with memory values B=1, 2, according to standard coding literature [11]. Issues regarding **Table 2** notation can be found in the cited reference. For purposes of simulation results analysis, it might suffice recalling the memory of involved codes. Before moving on to the analysis of simulation results we would like to point that all results should be interpreted under the assumption of Binary Symmetric Channel model with crossover probability $p_0 = \frac{1}{n}\sum_i p_i$.

[4] Decision tree with only one node

[5] R. Quinlan's decision tree algorithm in its C4.5 Revision 8 public version [10]

Table 2. RSSCs used in RECOC, Turbo learning with memory B

RSCC , B=1	RSCC , B=2
$\left[1 \quad \frac{1}{3}\right]$	$\left[1 \quad \frac{5}{7}\right]$

The RECOC_Turbo test error was measured against the maximum number I of turbo iterations and the number T of binary AdaBoost iteration steps, for memory values $B = 1,2$ and rates $r = \frac{1}{3}, \frac{1}{4}$. We denote by TB+AB+DS a simulation scheme involving the RECOC_Turbo algorithm based on the boosting version (AB) of a decision stump (DS) learner. The choice of DS binary learners was taken considering the standard boosting framework, which requires only weak learners. Alternatively, other binary learning algorithms like the binary C4.5 are also possible though not preferable in terms of complexity and memory requirements. Observed results show that RECOC_Turbo combined with inner binary boosting and a simple binary DS algorithm allows learning below C4.5 in some learning domains. It should be stressed, however, that in all cases the desired boosting effect in M valued output domain is always achieved. For the sake of brevity, in **Fig. 3** to **Fig. 8**, some representative 3D-views of RECOC_Turbo test error are shown. From simulations, one may expect that boosting regions may occur at low T - AdaBoost values and for a modest I number of turbo iteration steps, typically no more than 10. The results shown for *I*=0 turbo iterations represent the test error achieved when the decision is taken by looking only at the systematic part of a turbo codeword in the standard ECOC way. In addition, it was observed that higher B values do not contribute to enhancing test error responses if they could be obtained with lower ones. For Primary Tumor, Audiology and Glass learning domains, RECOC_Turbo improves learning capacity of the base Decision Stump algorithm in an M valued output domain but cannot learn below C4.5. This fact suggests that a strong binary learning scheme like a binary decision tree would be better a conjecture that was confirmed in practice. In addition, if RECOC_Turbo learning if achievable satisfactory at certain r^* value, then channel rates $r \le r^*$ do not improve test error responses because of the limited diversity than one can achieve with a unique and finite training sample. In addition, B values greater than 2, do not improve RECOC_Turbo test error (for the sake of brevity, we do not include these results but they were observed in practice). Furthermore, based on the concept of *interleaver gain* in theory of turbo codes (see [12], page 425), one may expect that better results would be obtained from higher interleaver lengths i.e. for higher M values. Therefore, RECOC_Turbo learning is also a promising alternative for higher M classification problems. Regarding comparisons with standard ECOC algorithms, we would like to stress that realizations of fair comparisons would require the implementation of Viterbi decoding of Turbo Codes, an approach which has been always avoided in recursive coding because of the inherent complexity [12].

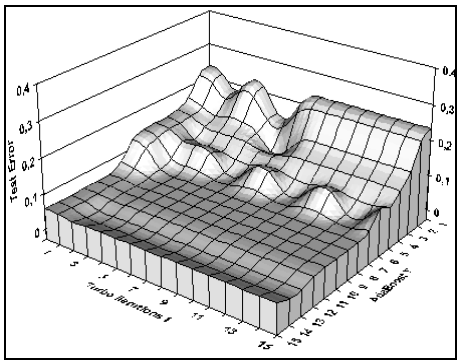

Fig. 3. Anneal, M=6. TB+AB+DS, B=1, $r = \frac{1}{3}$. Boosting can be achieved at almost all regions of the I-T plane

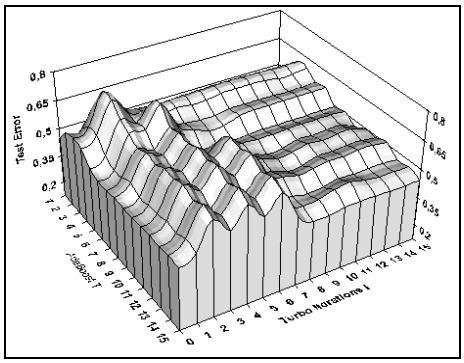

Fig. 4. Audiology, M=24. TB+AB+DS, B=1, $r = \frac{1}{3}$. Boosting is achieved with a small number I of turbo iterations

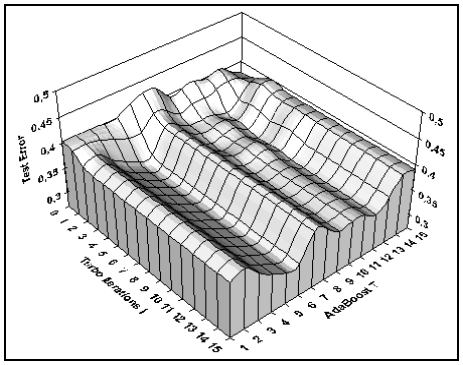

Fig. 5. Glass, M=7. TB+ AB+ DS, B=1, $r = \frac{1}{3}$. Boosting is achieved at low complexity regions of the I-T plane

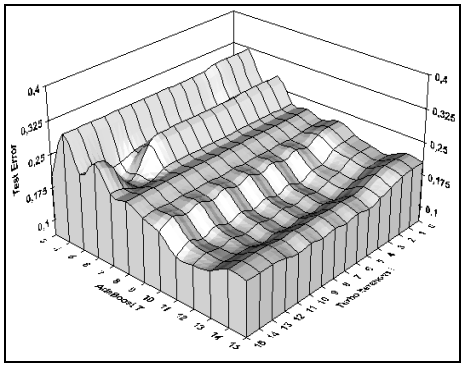

Fig. 6. Lymph, M=4. TB+AB+DS, B=2, $r = \frac{1}{3}$. Boosting is achieved at almost all regions of the I-T plane

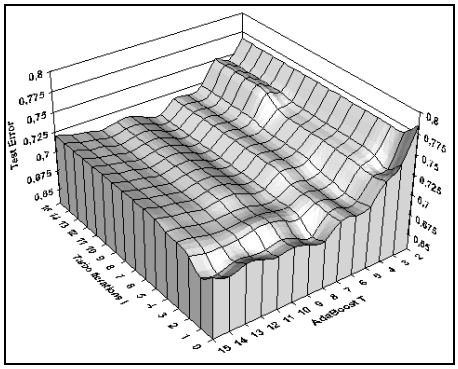

Fig. 7. Primary Tumor, M=22. TB+AB+DS, B=1, $r = \frac{1}{3}$. Boosting is achieved at almost all regions of the I-T plane

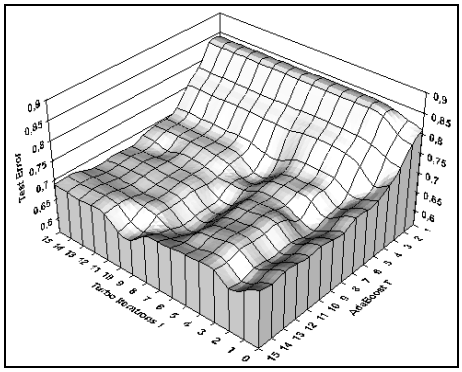

Fig. 8. Primary Tumor, M=22. TB+AB+DS, B=1, $r = \frac{1}{4}$. Boosting is improved with a smaller r value.

In **Table 3**, we compare the best RECOC_Turbo test error for fixed B=1, $r = \frac{1}{3}$ and DS learners against the test error achieved by ECOC learning when using the exhaustive coding schemes according to [2]. ECOC decoding is a Minimum L_1 distance decoding [2]. In fact, only the Primary Tumor domain seems to be sensitive to inner boosting under the standard ECOC framework with L_1 distance measure. In the other considered domains, RECOC_Turbo performs as well as or better than ECOC. It may be argued that other binary (and stronger) learning schemes like a binary C4.5 would perform better for the ECOC algorithm and this in fact true. However, for purposes of boosting it is well known that weak learning algorithms are preferred.

Table 3. Comparisons RECOC – ECOC test errors (66 % split) in M valued output domains. RECOC learning is analized under codewords lengths n_{TB}, I turbo decoding steps, T inner boosting steps, memory B and channel rate r. ECOC learning is analized under codewords lengths n_{ECOC}. In both cases an underlying Decision Stump learner is assumed.

Domain	M	n_{TB}	I	T	TB (B=1, $r = \frac{1}{3}$)	n_{ECOC}	ECOC
Anneal	6	12	5	2	0.06	31	1.0
Audiology	24	18	3	1	0.28	15	0.27
Glass	7	12	2	6	0.31	63	0.7
Lymph	4	9	0	13	0.18	7	0.96
Primary Tumor	22	18	1	8	0.66	15	0.57

6 Conclusions and Further Work

The main contribution of this work is the introduction of recursive related coding models for the design of low complexity classifiers in M valued output domains. We propose the design of ECOC like learning algorithms, which can take account of both binary learners' reliability and coding structure itself. This approach extends the binary boosting concept to M valued output domains in a transparent way. Furthermore, this formulation allows us to use AdaBoost under the ECOC framework because the improvement of base learners responses can always be mapped to improvements in the whole learning algorithm. We believe that this is the natural way to use AdaBoost and not in the converse. An important property arising from the introduced RECOC models is their ability to implement a generalized version of the diversity learning [14] concept, thus extending the concept of ensemble learning. In addition, this approach also addresses the design of ECOC schemes in arbitrary M valued output domains, a problem, which has remained opened.

A number of directions for further work and research stand out. Besides its validation with other UCI data domains (Java requires a careful use of memory) and real data, it remains to study convergence characteristics, the effect of channel model assumptions and alternative binary learning schemes. Finally, further algorithms can

be formulated from different underlying recursive codes [15] as well as alternative recursive decoding algorithms

References

1. Schapire, R. E., Singer, Y.: Improved Boosting Algorithms Using Confidence - rated Predictions. Machine Learning, Vol. 37, No. 3, pp. 277-296 (1999)
2. Dietterich, T., Bakiri, G.: Error-correcting output codes: A general method for improving multiclass inductive learning programs. Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91), pp. 572-577, Anaheim, CA: AAAI Press (1991)
3. Tanner, M.: A recursive Approach to Low Complexity Error Correcting Codes. IEEE Transactions on Information Theory, Vol. IT-27, pp. 533-547 (1981)
4. Berrou, C., Glavieux, A.: Near Optimum Error Correcting Coding and Decoding: Turbo Codes. IEEE Transactions on Communications, Vol. 44, No. 10, pp. 1261-1271 (1996)
5. Kschischang, F., Frey, B.: Iterative decoding of compound codes by probability propagation in graphical models. IEEE Journal on Selected Areas in Communications, Vol. 16-2, pp. 219-230 (1998)
6. Divsalar, D., Dolinar, S., Pollara, F.: Transfer Function Bounds on the Performance of Turbo Codes. TDA Progress Report 42-122 (1995).
7. Bahl L. R., Cocke J., Jelinek F., Rajiv J.: Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate. IEEE Transactions on Information Theory, Vol. 20, pp. 284-287 (1974)
8. Ryan, W.: A turbo code tutorial. Unpublished paper. http://www.ece.arizona.edu/~ryan/
9. Witten, I., Frank E.: Data Mining, Practical Machine Learning Tools and Techniques with JAVA Implementations. Morgan Kaufmann Publishers, San Francisco, California (2000)
10. Quinlan, R.: C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers (1993)
11. Benedetto, S., Divsalar, D., Montorsi, G., Pollara, F.: Soft-Output Decoding Algorithms in Iterative Decoding of Turbo Codes. TDA Progress Report 42-124 (1996)
12. Benedetto S., Montorsi G.: Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes. IEEE Transactions on Information Theory, Vol. 42, No. 2, pp. 409-428, (1996)
13. Benedetto, S., Montorsi, G.: Design of Parallel Concatenated Codes. IEEE Transactions on Information Theory, Vol. 44, No. 5, pp. 591- 600 (1996)
14. Dietterich, T.: Ensemble Methods in Machine Learning. In J. Kittler and F. Roli (Ed.) First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, pp. 1-15, New York: Springer Verlag (2000)
15. MacKay, D. J.: Good Error Correcting Codes based on Very Sparse Matrices. IEEE Trans. Inf. Theory, Vol. 45, pp. 399- 431 (1999)

Acknowledgements

Elizabeth Tapia's work is supported by an Argentinean Government FOMEC grant and the National University of Rosario, Argentina. Javier García's work is supported by the Amo Complutense Program and by the Spanish Ministry of Science and Technology (MCYT, Spain) under projects TIC2000-0735 and TIC2000-0737-C03-02. During this work he was with the IBM Research Division K65/C2 at IBM Almaden Research Center, San Jose, California, USA (javiervi@almaden.ibm.com).