Compact Evolutive Design of Error-Correcting Output Codes

Miguel Àngel Bautista[1], Xavier Baró[1,2,3], Oriol Pujol[1,2], Petia Radeva[1,2], Jordi Vitrià[1,2], and Sergio Escalera[1,2]

[1]Dept. Matemàtica Aplicada i Anàlisi, Gran Via les Corts Catalanes 585, Barcelona
[2]Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Barcelona
[3]Department of Computer Science, Multimedia, and Telecomunications, Universitat Oberta de Catalunya, Rambla del Poblenou 156, 08018, Barcelona
miguelangelbautistamartin@gmail.com,
{xevi, oriol, petia, jordi, sergio}@maia.ub.es

Abstract. The classification of large number of object categories is a challenging trend in the Machine Learning field. In literature, this is often addressed using an ensemble of classifiers. In this scope, the Error-Correcting Output Codes framework has demonstrated to be a powerful tool for the combination of classifiers. However, most of the state-of-the-art ECOC approaches use a linear or exponential number of classifiers, making the discrimination of a large number of classes unfeasible. In this paper, we explore and propose a minimal design of ECOC in terms of the number of classifiers. Evolutionary computation is used for tuning the parameters of the classifiers and looking for the best Minimal ECOC code configuration. The results over several public UCI data sets and a challenging multi-class Computer Vision problem show that the proposed methodology obtains comparable and even better results than state-of-the-art ECOC methodologies with far less number of dichotomizers.

Keywords: Ensemble of Dichotomizers, Error-Correcting Output Codes, Evolutionary optimization

1 Introduction

Nowadays challenging applications of Machine Learning deal with changing environments, online adaptations, contextual information, etc. In order to deal with all these problems, efficient ways for processing huge amount of data are often required. Usual machine learning strategies are effective for dealing with small number of classes. The choices are limited when the number of classes becomes large. In that case, the natural algorithms to consider are those that model classes in an implicit way, such as instance based learning (i.e. nearest neighbors). However, this choice is not necessarily the most adequate for a given problem. Moreover, we are forgetting many algorithms of the literature such as ensemble learning (i.e. Adaboost) or kernel based discriminant classifiers (i.e. support vector machines) that have been proven to be very powerful tools.

Most of state-of-the-art multi-class architectures need to deal with the discrimination of each class either by modeling its probability density function, or by storing a classification boundary and using some kind of aggregation/selection

function to obtain a final decision. Another way to deal with this kind of problems is to use a divide-and-conquer approach, such as flat strategies (voting), hierarchical classifiers, or Error-Correcting Output Codes (ECOC). ECOC encodes different partitions of the problem in a matrix of codewords (one codeword per class) and the final decision is obtained by looking at the most similar codeword at the test step. ECOC allows the inclusion of flat strategies as well as hierarchical classifiers [1]. Moreover, the analysis of the ECOC error evolution has demonstrated that ECOC corrects errors caused by the bias and the variance of the learning algorithm [2]. However, note that by construction or in order to obtain the desired performance, most of the strategies need between N and N^2 classifiers, given N different classes. Although this is adequate and acceptable when the number of classes is small, it becomes prohibitive when the number of classes becomes large. This number of classifiers has been recently reduced in some ECOC designs, such as the DECOC approach of [1], that requires $N-1$ classifiers. The Dense Random and Sparse Random designs also reduce this number of classifiers to $15 \cdot log_2(N)$ and $10 \cdot log_2(N)$, respectively. However this kind of approaches design the problems without taking into account the underlying distribution of the class characteristics.

The goal of this paper is to propose and evaluate different general ways of making the multi-class machine learning problem tractable when the number of categories makes most of the models computationally unfeasible. In particular, we are interested in methods that scale sub-linearly with the number of classes, allowing their applicability in general Machine Learning problems. The proposal relies on the Error Correcting Output Codes framework, reducing the number of binary classifiers that have to be trained in the ensemble. Following the Occam razor principle, we propose a minimal ECOC design of size $log_2(N)$ in terms of the number of classifiers. An evolutionary approximation, similar to the one proposed in [3] is proposed for tuning the parameters of the classifiers and looking for a Minimal design with high generalization capabilty. Moreover, this design is problem dependent in the sense that the evolved ECOC fits the distribution of the object characteristics. The novel Minimal ECOC is compared with the state-of-the-art ECOC approaches, obtaining comparable and even better results when classifying several object categories in different Machine Learning applications with far less cost.

The paper is organized as follows: Section 2 presents the Minimal ECOC design. Section 3 evaluates the novel methodology comparing with the state-of-the-art approaches on public and challenging Pattern Recognition Applications. Finally, Section 4 concludes the paper.

2 Minimal Error-Correcting Output Codes

In this section, we review the ECOC framework and propose a Minimal ECOC design in terms of the number of classifiers.

2.1 Error-Correcting Output Codes

Given a set of N classes to be learnt in an ECOC framework, n different bi-partitions (groups of classes) are formed, and n binary problems (dichotomizers)

over the partitions are trained. As a result, a codeword of length n is obtained for each class, where each position (bit) of the code corresponds to a response of a given dichotomizer (coded by +1 or -1 according to their class set membership). Arranging the codewords as rows of a matrix, we define a *coding matrix* M, where $M \in \{-1, +1\}^{N \times n}$ in the binary case. In Figure 1 we show an example of a binary coding matrix M. The matrix is coded using five dichotomizers $\{h_1, ..., h_5\}$ for a 4-class problem $\{c_1, ..., c_4\}$ of respective codewords $\{y_1, ..., y_4\}$. The hypotheses are trained by considering the labeled training data samples $\{(\rho_1, l(\rho_1)), ..., (\rho_m, l(\rho_m))\}$ for a set of m data samples. The white and black regions of the coding matrix M are coded by +1 and -1, respectively. For example, the first classifier is trained to discriminate c_3 against c_1, c_2, and c_4; the second one classifies c_2 and c_3 against c_1 and c_4, etc., as follows:

$$h_1(x) = \begin{cases} 1 & \text{if } x \in \{c_3\} \\ -1 & \text{if } x \in \{c_1, c_2, c_4\} \end{cases}, \ldots, \quad h_5(x) = \begin{cases} 1 & \text{if } x \in \{c_2, c_4\} \\ -1 & \text{if } x \in \{c_1, c_3\} \end{cases} \tag{1}$$

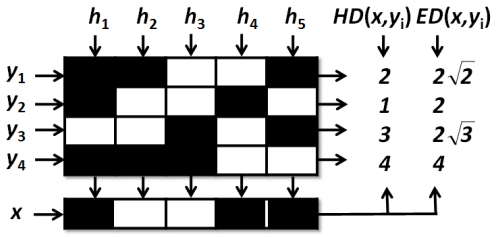

Fig. 1. Binary ECOC design for a 4-class problem. An input test codeword x is classified by class c_2 using the Hamming or the Euclidean Decoding.

The standard binary coding designs are the one-versus-all [4] strategy with N dichotomizers and the dense random strategy [5], with $10 \log_2 N$ classifiers. In the case of the ternary symbol-based ECOC, the 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 for a given classifier. In this ternary framework, the standard designs are the one-versus-one strategy [6] and the sparse random strategy [5], with $\frac{N(N-1)}{2}$ and $15 \log_2 N$ binary problems, respectively.

During the decoding process, applying n binary classifiers, a code x is obtained for each data sample ρ in the test set. This code is compared to the base codewords $(y_i, i \in [1, .., N])$ of each class defined in the matrix M, and the data sample is assigned to the class with the *closest* codeword. In Figure 1, the new code x is compared to the class codewords $\{y_1, ..., y_4\}$ using Hamming [4] and Euclidean Decoding [5]. The test sample is classified by class c_2 in both cases, correcting one bit error.

In literature there roughly exists three different lines for decoding [7]: those based on similarity measurements, including the Hamming and Euclidean decoding, probabilistic approaches, and loss-functions strategies.

2.2 Minimal ECOC Coding

Although the use of large codewords was initially suggested in order to correct as many errors as possible at the decoding step, high effort has been put into improving the robustness of each individual dichotomizer so that compact codewords can be defined in order to save time. In this way, the one-versus-all ECOC coding has been widely applied for several years in the binary ECOC framework. Although the use of a reduced number of binary problems often implies dealing with more data per classifier (i.e. compared to the one-versus-one coding), this approach has been defended by some authors in the literature demonstrating that the one-versus-all technique can reach comparable results to the rest of combining strategies if the base classifier is properly tuned [8]. Recently, this codeword length has been reduced to $N-1$ in the DECOC approach of [1], where the authors codify $N-1$ nodes of a binary tree structure as dichotomizers of a ternary problem-dependent ECOC design. In the same line, several problem-dependent designs have been recently proposed [9, 1, 10]. The new techniques are based on exploiting the problem domain by selecting the representative binary problems that increase the generalization performance while keeping the code length "relatively" small.

Although one-versus-all, DECOC, dense, and sparse random approaches have a relatively small codeword length, we can take advantage of the information theory principles to obtain a more compact definition of the codewords. Having a N-class problem, the minimum number of bits necessary to codify and univocally distinguish N codes is $B = \lceil \log_2 N$, where $\lceil . \rceil$ rounds to the upper integer.

For instance, we can think in a codification where the class codewords correspond to the N first Gray or binary code sequences of B bits, defining the Gray or binary Minimal ECOC designs. Note that this design represents the minimal ECOC codification in terms of the codeword length. An example of a binary Minimal ECOC, Gray Minimal ECOC, and one-versus-all ECOC designs for a 8-class problem are shown in Figure 2. The white and black positions correspond to the symbols +1 and -1, respectively.

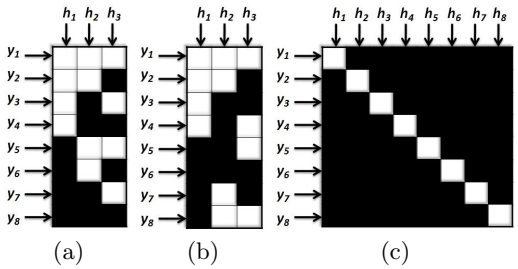

Fig. 2. (a) Binary Minimal, (b) Gray Minimal, and (c) one-versus-all ECOC coding designs of a 8-class problem.

Besides exploring predefined binary or Gray minimal coding matrices, we also propose the design of different minimal codification based on the distribution of the data and the characteristics of the applied base classifier, which can

increase the discrimination capability of the system. However, finding a suitable minimal ECOC matrix for an $N-$class problem requires to explore all the possible $N \times B$ binary matrices, where B is the minimum codeword length in order to define a valid ECOC matrix. For this reason, we also propose an evolutionary parametrization of the Minimal ECOC design.

Evolutionary Minimal Parametrization When defining a minimal design of an ECOC, the possible lost of generalization performance has to be taken into account. In order to deal with this problem an evolutionary optimization process is used to find a minimal ECOC with high generalization capability.

In order to show the parametrization complexity of the Minimal ECOC design, we first provide an estimation of the number of different possible ECOC matrices that we can build, and therefore, the search space cardinality. We approximate this number using some simple combinatorial principles. First of all, if we have an $N-$class problem and B possible bits to represent all the classes, we have a set CW with 2^B different words. In order to build an ECOC matrix, we select N codewords from CW without reposition. That is, taking N from a variation of 2^B elements and considering the symmetry of binary partitions, we can construct $V_N^{2^B} = \frac{2^B!}{2N(2^B-N)!}$ different ECOC matrices.

In this type of scenarios evolutionary approaches are often introduced with good results. Evolutionary algorithms are a wide family of methods that are inspired on the Darwin's evolution theory, and used to be formulated as optimization processes where the solution space is not differentiable or is not well defined. In these cases, the simulation of natural evolution process using computers results in stochastic optimization techniques which often outperform classical methods of optimization when applied to difficult real-world problems. Although the most used and studied evolutionary algorithms are the Genetic Algorithms (GA), from the publication of the *Population Based Incremental Learning* (PBIL) in 1995 by Baluja and Caruana [11], a new family of evolutionary methods is striving to find a place in this field. In contrast to GA, those new algorithms consider each value in the chromosome as a random variable, and their goal is to learn a probability model to describe the characteristics of good individuals. In the case of PBIL, if a binary chromosome is used, a uniform distribution is learnt in order to estimate the probability of each variable to be one or zero.

In this paper we experiment with both evolutionary strategies, GA and PBIL.

Problem encoding: The first step in order to use an evolutionary algorithm is to define the problem encoding, which consists of the representation of a certain solution or point in the search space by means of a *genotype* or alternatively a *chromosome* [12]. Binary encoding is the most common, mainly because first works about GA used this type of encoding. In binary encoding, every chromosome is a string of bits 0 or 1. Each ECOC is encoded as a binary chromosome $\zeta =< h_1^{c_1}, \ldots, h_B^{c_1}, h_1^{c_N}, \ldots, h_B^{c_N} >$, where $h_i^{c_j} \in \{0, 1\}$ is the expected value of the $i-th$ classifier for the class c_j, which corresponds to the $i-th$ bit of the class c_j codeword.

Adaptation function: Once the encoding is defined, we need to define the adaptation function, which associates to each individual its adaptation value to the environment, and thus, their survivor probability. In the case of the ECOC framework, the adaptation value must be related to the classification error.

Given a chromosome $\zeta =< \zeta_0, \zeta_1, \ldots, \zeta_L >$ with $\zeta_i \in \{0, 1\}$, the first step is to recover the ECOC matrix M codified in this chromosome. The values of M allows to create binary classification problems from the original multi-class problem, following the partitions defined by the ECOC columns. Each binary problem is addressed by means of a binary classifier, which is trained in order to separate both partitions of classes. Assuming that there exists a function $y = f(\mathbf{x})$ that maps each sample $\mathbf{x}$ to its real label y, to train a classifier means to find the best parameters w^* of a certain function $y = f'(\mathbf{x}, \mathbf{w})$, in the manner that for any other $\mathbf{w} \neq \mathbf{w}^*$, $f'(\mathbf{x}, \mathbf{w}^*)$ is a better approximation to f than $f'(\mathbf{x}, \mathbf{w})$. Once $\mathbf{w}^*$ are estimated for each binary problem, the adaptation value corresponds to the classification error. In order to take into account the generalization power of the trained classifiers, the estimation of $\mathbf{w}^*$ is performed on a subset of samples, while the rest of the samples are reserved for validation. The adaptation value for an individual represented by a certain chromosome ζ_i can be formulated as:

$$\varepsilon_i(X, Y, M_i) = \sum_j \delta_j(x_j, M_i \neq y_j) \tag{2}$$

where M_i is the ECOC matrix encoded in ζ, $X = \langle x_1, \ldots, x_N \rangle$ a set of samples, $Y = \langle y_1, \ldots, y_N \rangle$ the expected labels for samples in X, and δ_i is the function that returns the classification label applying the decoding strategy.

Evolutionary process: Once the encoding and adaptation function have been defined, we use standard implementation for GA and PBIL, in order to evolve the Minimal ECOC matrices. In the case of GA, scattered crossover operator is used, in which, we generate a random binary vector, with a binary value assigned to each gene. The first child is created using all the genes from the first parent in those positions with a value of one, and the genes of the second parent with positions with the value zero. The second child is created as the complementary of the first one. That is, taking genes from second parent for values one and from first parent for values zero. In order to introduce variations to the individuals, we use mutation operator that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped if it falls outside of the user-specified lower or upper bounds for that gene. For PBIL, the best two individuals of each population are used to update the probability distribution. At each generation, this probability distribution is used to sample a new population of individuals. A uniform random noise is applied to the probability model to avoid convergence to local minima.

Finally, in both evolutionary strategies we adopt an *Island Model* evolution scheme in order to exploit a more coarse grain parallel model. The main idea is to split a population of K individuals into S sub-populations of K/S individuals. If each sub-population is evolved independently from the others, genetic drift will

tend to drive these populations in different directions. By introducing migration, the *Island Model* is able to exploit differences in the various sub-populations (this variation in fact represents a source of genetic diversity). Each sub-population is an island and there is a chance movement of genetic material from one island to another.

Training the binary classifiers: In [8], Rifkin concludes that the number of classifiers in the ECOC problem can be reduced using more accurate classifiers. Therefore, in this paper we adopt the Support Vector Machines with Gaussian Radial Basis Functions kernel (SVM-RBF). Training a SVM often implies the selection of a subset of data points (the support vectors), which are used in order to build the classification boundaries. In the specific case of Gaussian RBF kernels, we need to optimize the kernel parameter γ and the regularizer C, which have a close relation to the data distribution. While the support vectors selection is part of the SVM learning algorithm, and therefore, is clearly defined, finding the best C and γ is addressed in literature with various heuristic or brute-force approaches. The most common approach is the use of cross-validation processes which select the best pair of parameters for a discretization of the parameters space. Nevertheless, this can be viewed as another optimization problem. An therefore, it can be faced using evolutionary algorithms. For each binary problem, defined by one column of the ECOC matrix, we use Genetic Algorithms in order to find good values for C and γ parameters, using the same settings than in [3], where individuals correspond to a pairs of genes, and each gene corresponds to the binary codification of a floating point value.

3 Results

In order to present the results, first, we discuss the data, methods, and evaluation measurements of the experiments.

• *Data*: The first data used for the experiments consists of twelve multi-class data sets from the UCI Machine Learning Repository database [13]. Then, we apply the classification methodology in the public *Labeled Faces in the Wild* [14] data set to perform the multi-class face classification of a large problem consisting of 610 face categories.

• *Methods*: We compare the one-versus-one [6] and one-versus-all [4] ECOC approaches with the binary and evolutionary Minimal approaches. For simplicity we omitted the Gray Minimal design. The Hamming decoding is applied at the decoding step [15]. The ECOC base classifier is the OSU implementation of SVM with Radial Basis Function kernel [16]. The SVM C and γ parameters are tuned via Genetic Algorithms and PBIL for all the methods, minimizing the classification error of a two-fold evaluation over the training sub-set.

• *Evaluation measurements*: The classification performance is obtained by means of a stratified ten-fold cross-validation, and testing for the confidence interval with a two-tailed t-test. We also apply the Friedman test [17] in order to look for statistical significance among the obtained performances.

3.1 UCI categorization

The classification results obtained for all the UCI data sets considering the different ECOC configurations are shown in Table 1. In order to compare the performances provided for each strategy, the table also shows the mean rank of each ECOC design considering the twelve different experiments. The rankings are obtained estimating each particular ranking r_i^j for each problem i and each ECOC configuration j, and computing the mean ranking R for each design as $R_j = \frac{1}{N}\sum_i r_i^j$, where N is the total number of data sets. We also show the mean number of classifiers (#) required for each strategy.

Table 1. UCI classification results.

	Binary Minimal ECOC		Evol. Minimal ECOC		one-vs-all ECOC		one-vs-one ECOC	
Data set	Perf.	Classif.	Perf.	Classif.	Perf.	Classif.	Perf.	Classif.
Derma	96.0±2.9	3	**96.3±2.1**	3	95.1±3.3	6	94.7±4.3	15
Iris	96.4±6.3	2	**98.2±1.9**	2	96.9±6.0	3	96.3±3.1	3
Ecoli	80.5±10.9	3	**81.4±10.8**	3	79.5±12.2	8	79.2±13.8	28
Vehicle	72.5±14.3	2	76.99±12.4	2	74.2±13.4	4	**83.6±10.5**	6
Wine	95.5±4.3	2	**97.2±2.3**	2	95.5±4.3	3	**97.2±2.4**	3
Segment	96.6±2.3	3	96.6±1.5	3	96.1±1.8	7	**97.18±1.3**	21
Glass	56.7±23.5	3	50.0±29.7	3	53.85±25.8	6	**60.5±26.9**	15
Thyroid	**96.4±5.3**	2	93.8±5.1	2	95.6±7.4	3	96.1±5.4	3
Vowel	57.7±29.4	3	**81.78±11.1**	3	80.7±11.9	8	78.9±14.2	28
Balance	80.9±11.2	2	87.1±9.2	2	89.9±8.4	3	**92.8±6.4**	3
Shuttle	80.9±29.1	3	83.4±15.9	3	**90.6±11.3**	7	86.3±18.1	21
Yeast	50.2±18.2	4	**54.7±11.8**	4	51.1±18.0	10	52.4±20.8	45
Rank & #	2.9	**2.7**	**2.0**	**2.7**	2.7	5.7	2.2	15.9

In order to analyze if the difference between method ranks is statistically significant, we apply a statistical test. In order to look if the measured ranks differ from the mean rank we use the Friedman test. The Friedman statistic value is computed as $X_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$. In our case, with $k = 4$ ECOC designs to compare, $X_F^2 = -4.94$. Since this value is undesirable conservative, Iman and Davenport proposed a corrected statistic: $F_F = \frac{(N-1)X_F^2}{N(k-1)-X_F^2}$. Applying this correction we obtain $F_F = -1.32$. With four methods and twelve experiments, F_F is distributed according to the F distribution with 3 and 33 degrees of freedom. The critical value of $F(3, 33)$ for 0.05 is 2.89. As the value of F_F is no higher than 2.98 we can state that there are no statistical different among the ECOC performances. This means that all four strategies are suitable in order to deal with multi-class categorization problems. This result is very satisfactory and encourages the use of the Minimal approach since similar (or even better) results can be obtained with far less number of classifiers. Moreover, the GA evolutionary version of the Minimal design improves in the mean rank to the rest of classical coding strategies, and in most cases outperforms the binary Minimal approach in the present experiment. This result is expected since the evolutionary version looks for a minimal ECOC matrix configuration that minimizes the error over the training data. In particular, the advantage of the evolutionary version over the binary one is more significant when the number of classes increases, since more minimal matrices are explored, and hence, on an average, it is capable of finding a better solution.

On the other hand, possible reasons why the evolutionary Minimal ECOC design obtains similar or even better performance results than the one-versus-one and one-versus-all approaches with far less number of dichotomizers can be

the few classifiers considered for tuning, and the use of all the classes in balanced binary problems, which can help the system to increase generalization if a good decision boundary can be found by the classifier. Note that the one-versus-one classifier looks for binary problems that split just two classes. In those cases, though good and fast solutions could be found in training time, the use of less data does not assure a high generalization capability of the individual classifiers.

In terms of testing time, since all the trained classifiers spend the same time for testing, classification time is proportional to the number of trained classifiers. The mean number of dichotomizers used for each strategy is shown in the last row of Table 1. Observe the great difference in terms of the number of classifiers between the minimal approaches and the classical ones. The Minimal approaches obtain an average speed up improvement of 111% respect the one-versus-all approach in testing time. Meanwhile in the case of the one-versus-one technique this improvement is of 489%.

In the next section we test if the same behavior occurs classifying a challenging Computer Vision problem with several object categories.

3.2 Labeled Faces in the Wild categorization

This data set contains 13000 faces images taken directly from the web from over 1400 people. This images are not constrained in terms of pose, light, occlusions or any other relevant factor. For the purpose of this experiment we used a specific subset, taking only the categories which at least have four or more examples, having a total of 610 face categories. Finally, in order to extract relevant features from the images, we apply an Incremental Principal Component Analysis procedure [18], keeping 99.8% of the information. An example of face images is shown in 3.

Fig. 3. Labeled Faces in the Wild data set.

The results in Table 2 show that the best performance is obtained by the Evolutionary GA and PBIL Minimal strategies. One important observation is that Evolutionary strategies outperform the classical one-versus-all approach, with far less number of classifiers (10 instead of 610). Note that in this case we omitted the one-vs-one strategy since it requires 185745 classifiers for discriminating 610 face categories.

Table 2. Labeled Faces in the Wild data set classification results.

	Binary M. ECOC		GA M. ECOC		PBIL M. ECOC		one-vs-all		one-vs-one	
Data set	Perf.	#	Perf.	#	Perf.	#	Perf.	#	Perf.	#
FacesWild	26.4±2.1	10	**30.7±2.3**	10	30.02.4±	10	25.0±3.1	610	-	185745

4 Conclusion

We presented a general methodology for the classification of several object categories which only requires $\lceil \log_2 N \rceil$ classifiers for a N-class problem. The methodology is defined in the Error-Correcting Output Codes framework, designing a minimal coding matrix in terms of dichotomizers which univocally distinguish N codes. Moreover, in order to speed up the design of the coding matrix and the tuning of the classifiers, evolutionary computation is also applied.

The results over several public UCI data sets and a challenging multi-class Computer Vision problem with several object categories show that the proposed methodology obtains statistically equivalent results as the state-of-the-art ECOC methodologies with far less number of dichotomizers. For example, the Minimal approach trained 10 classifiers to split 610 face categories, meanwhile the one-versus-all and one-versus-one approaches required 610 and 185745 dichotomizers, respectively.

References

1. O. Pujol, P. Radeva, J. Vitrià, Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes, in: Trans. on PAMI, Vol. 28, 2006, pp. 1001–1007.
2. T. Dietterich, E. Kong, Error-correcting output codes corrects bias and variance, in: ICML (Ed.), S. Prieditis and S. Russell, 1995, pp. 313–321.
3. A. C. Lorena, A. C. de Carvalho, Evolutionary tuning of svm parameter values in multiclass problems, Neurocomputing 71 (16-18) (2008) 3326 – 3334.
4. M. Pelikan, D. E. Goldberg, Learning machines, in: McGraw-Hill, 1965.
5. E. Allwein, R. Schapire, Y. Singer, Reducing multiclass to binary: A unifying approach for margin classifiers, in: JMLR, Vol. 1, 2002, pp. 113–141.
6. T.Hastie, Classification by pairwise grouping, NIPS 26 (1998) 451–471.
7. S. Escalera, O. Pujol, P. Radeva, On the decoding process in ternary error-correcting output codes, PAMI 99 (1).
8. R. Rifkin, A. Klautau, In defense of one-vs-all, JMLR 5 (2004) 101–141.
9. K. Crammer, Y. Singer, On the learnability and design of output codes for multiclass problems, in: Machine Learning, Vol. 47, 2002, pp. 201–233.
10. S. Escalera, O. Pujol, P. Radeva, Error-correcting output codes library, Journal of Machine Learning Research 11 (2010) 661–664.
11. S. Baluja, R. Caruana, Removing the genetics from the standard genetic algorithm, in: ICML, 1995, pp. 38–46.
12. J. Holland, Adaptation in natural and artificial systems: An analysis with applications to biology, control, and AI, University of Michigan Press, 1975.
13. A. Asuncion, D. Newman, UCI machine learning repository, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.
14. G. B. Huang, M. Ramesh, T. Berg, E. L. Miller, Labeled faces in the wild, Tech. Rep. University of Massachusets Amherst, 07-49 (October 2007).
15. T. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, in: JAIR, Vol. 2, 1995, pp. 263–286.
16. OSU-SVM-TOOLBOX, http://svm.sourceforge.net/.
17. J. Demsar, Statistical comparisons of classifiers over multiple data sets, JMLR 7 (2006) 1–30.
18. W. Hwang, J. Weng, Y. Zhang, Candid covariance-free incremental principal component analysis, PAMI 25 (8) (2003) 1034–1040.