Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

SUBLINEAR EVOLUTIONARY DESIGN OF ERROR CORRECTING OUTPUT CODES

Miguel Angel Bautista Martin

Advisor/s: Dr. Sergio Escalera Guerrero
Dr./Drs. on behalf of the Advisor/s: Dr. Xavier Baró i Sole

14/06/2010

Abstract

One of the first junctures in artificial intelligence and machine learning is to increase the interactivity of intelligent systems with their environment. Most of the actual strategies use sensors in order to simulate the data acquisition process that we obtain by means of our senses. Once the information has been extracted, one of the main challenges consists of developing suitable techniques for the processing of huge amount of data. The aim of this process is to take a decision between a wide set of possibilities. In this report, we propose a new methodology capable of processing huge sets of data, being able to discriminate between a wide set of categories. In particular, we focus on the error correcting output codes framework, defining a logarithmic number of classifiers. Genetic algorithms are used to define an optimum subset of problems as well as to find the optimum parameters that increase the generalization capability of the classifiers. The approach is tested over a wide number of the public machine learning and computer vision community datasets, getting results that outperform state-of-the-art strategies, at the same time that significantly reduces the computational cost.

Contents

Chapter 1

Introduction and Motivation

1.1 Introduction

Nowadays challenging applications of Pattern Recognition 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. One clear example is the case of general Pattern Recognition algorithms for classification, especially when the number of categories, namely objects, people, brands, etc, is arbitrarily large. 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 [FHT98]) or kernel based discriminant classifiers (i.e. support vector machines [SVM03]) 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 modelling 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. Instead of extending a method to cope with the multi-class case, one can divide the multi-class problem into smaller binary problems and then combine their responses using some kind of committee strategy, such as voting. In literature, one can roughly find three main lines of research in the last tendency: flat strategies, hierarchical classification, and Error Correcting Output Codes (ECOC). Flat strategies consist of using some predefined problem partition followed by some kind of voting strategy. Good examples of this line are strategies like one-against-all or all-

pairs. Hierarchical classification relies on some similarity metric among classes for creating a binary tree in which at each node a particular partition of the classes is considered. Finally, 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 can be regarded as a generalization of the former strategies since it allows the inclusion of flat strategies as well as hierarchical classifiers [PRV06]. 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 [DK95][1]. 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 [PRV06], 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 underlaying distribution of the class characteristics.

On the other hand, since the very beginning evolution and genetics are solving the problem of adapting living beings to the environment in which they usually can be found. Evolution can be seen from a computational point of view as a tool for finding solutions for a given problem in way which is not exhaustive neither analytical. This way of treating the problem is based on the theorem proposed by [Dar], in which the argumentation of how the adaptation to the environment performed by a living being improves with the flow of generations. In this sense, we could envisage an algorithm which aims to solve a problem, treating it as an evolutive process in which a set of random possible solutions for the problem are mixed in a strategic way, in order to improve the adaptation to the environment, therefore, obtaining better solutions.

1.2 Definition of the problem and goals

As mentioned before, one of the main research fields of AI is to improve the interactivity of expert or intelligent systems with their environment. In order to reach this higher interactivity, a common approach is the use of sensors to retain as much information of the environment as possible. Therefore, the major breakthrough of IA is to find a way to process this huge amount of information that the sensors are capable to obtain.

Very often the process of this information implies the classification or categorization of the data, where the number of classes is arbitrarily big.

[1] The bias term describes the component of the error that results from systematic errors of the learning algorithm. The variance term describes the component of the error that results from random variation and noise in the training samples and random behavior of the learning algorithm. For more details, see [DK95].

The goal of this thesis is to propose and evaluate different general ways of making the multi-class pattern recognition 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 Pattern Recognition 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, is proposed for tuning the parameters of the classifiers and looking for a Minimal design with high generalization capability. 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 (even better results) when classifying several object categories in different Pattern Recognition applications with far less cost.

1.3 State of the art on multi-class categorization problems

Multi-classification problems have been a field of study of various researchers during the past years, without achieving a solution both clear and efficient. In this section we analyze the works that deal with this kind of problems.

1.3.1 Artificial Neural Networks

In the past fifty years the application of ANN has been deeply studied and one of the most applied architectures was the *Multi-layer Perceptron* (MLP). This architecture is based in the principal of usual neural networks. The MLP consists at least of 3 fully connected layers, the input layer, the hidden layer and the output layer. The reason of the use of this architecture is that, the MLP is the minimal network which is a universal approximator of any function.

In any ANN there are three main parts:

- The neurons, which are the main units of process.
- The connection, which are the edges that connect neurons.
- The weights, which are the importance of each connection.

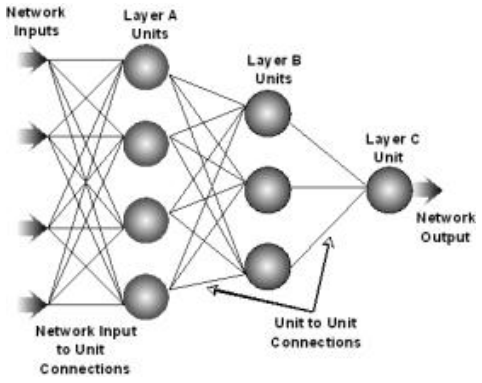


Figure 1.1: Multi-layer Perceptron

For a MLP the type of output that generates the network is of the following kind:

$$y_k(x) = g(\sum_{i=0}^{W} w_{xi}\phi_i(x)) \tag{1.1}$$

$$\phi_0(x) = 1 \tag{1.2}$$

$$\phi_i(x) = g(\sum_{j=0}^{n} v_{ij}x_j) \tag{1.3}$$

$$x_0 = 1 \tag{1.4}$$

To learn what are the sub-optimal weights for the ANN, usually the *back-propagation* algorithm. Which is an gradient descent based algorithm and as a matter of fact it inherits all the properties of the gradient descend, such as, finding sub-optimal results and convergence to local optima. In fact this algorithm gives us the update rule for every node in the network. Let's consider the error produced in the output node k in the $n-th$ data point as o $e_k(n) = o_k(n) - p_k(n)$, where $p_k(n)$ is the predicted value of the perceptron and $o_k(n)$ is the real target value of the $n-th$ data point. Our goal is to minimize the energy of the output error given by:

$$\varepsilon(n) = \frac{1}{2}\sum e_k^2(n) \tag{1.5}$$

By differential theory it can be demonstrated that the update rule is

$$\triangle w_{ki}(n) = -\eta\frac{\partial\varepsilon(n)}{\partial v_k(n)}p_i(n) \tag{1.6}$$

Where $p_i(n)$ is the value of the previous neuron and η is the learning rate. Further analysis is very complex and is out of the scope of the thesis, as a final

simplification it can be seen that in order to update the weight of a layer k we must first update the weights in layer $k-1$ with the previous updating rule. Therefore this algorithm can be seen as the backpropagation of the error made by every neuron.

1.3.2 Classification Trees

A Classification Tree is a predictive model which analyzes observations over an item to reach some conclusion of its value. In this kind of trees, leaves represent classifications and branches represent sets of features which have lead to this classifications. Every leaf represents a value of the target variable calculated by means of input variables (interior nodes) in the path from the leaf to the root.

The tree construction algorithms are based in choosing the feature of the data that better allows the separation of the data in better subsets according to the target value, in other words, the entropy of data is used, although another metrics could be used. The selected feature to build the tree would b as good as the separation of the data that it could do, according to the target variable.

One of the most used algorithms for this calculus is the *information gain* algorithm.

$$I_e(f) = -\sum f_i \log_2 f_i \tag{1.7}$$

The most common algorithm in classification trees construction is the Iterative Dichotomizer 3, which in a intuitive form will be presented as the following.

1. Take all the features and calculate its entropy with relation of the test set.
2. Choose the feature with maximum entropy, in other word, that maximizes the information gain
3. Make the node of tree to contain this feature

1.3.3 Ensemble approaches

Mixture Of Experts

An approach with some similarities to Classification Trees is the Mixture Of Experts model. It's based on the idea that humans usually do when dealing with a great complexity decision. For example, a doctor before diagnosing a terminal disease, such as cancer, consults with various specialist on the particular theme, such as oncologist and also before taking the final decision the doctor will have the patient severally tested to make sure of the correctness of the diagnose. Therefore, the mixture of expert model will behave on the same way.

This model contains a set of classifiers $C_1, ..., C_t$ which will constitute the *ensemble*. The a second level classifier C_{t+1} will assign weights every classifier of the ensemble. The C_{t+1} is also known as gated network because usually this classifier is a ANN trained by *expectation minimization*. This weighted outputs will go through a combination system that will give the final result.

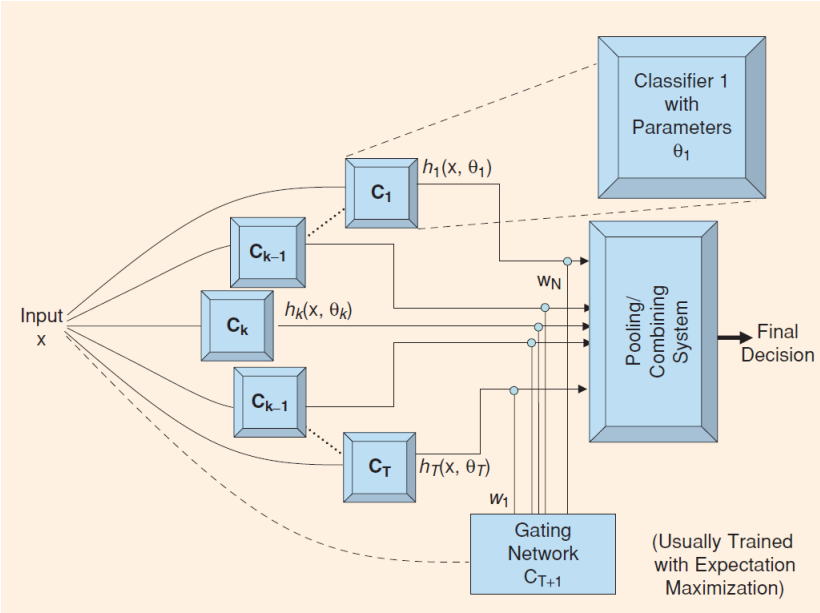


Figure 1.2: Mixture Of Experts model

As we can see in the figure 1.2, the model can be seen as a classification algorithm where each first level classifier is an expert on a space of features of the input sample x. The gated network will choose by means of weights, the classifiers or the subset of classifiers most appropriated to each input sample. Finally to generate the final prediction, the model can combine the weighted output in several ways, for example, calculating the weighted sum of all the ensemble or choosing the heaviest output or depending on the output type of the experts (discrete or real).

Error Correcting Output Codes

In the last decade some studies have shown that represent the multi-class problem in the ECOC framework represents significant improvement when dealing with problem dependent designs. The ECOC framework has been widely applied in classical AI problems such as facial recognition [KGWM01] or hand-write recognition [Gha02].

An Error Correcting Output Code can be seen as a matrix in which the rows denote the classes to discriminate in our system and the columns to the hypothesis which we have to distinguish those classes.

The ECOC framework consist mainly of two steps: in the first one, called coding a codeword is assign to each class so it is univocally distinguish from the other classes. On the second step, called decoding, each sample to classify is compared to all codewords and a prediction of classification is obtained.

One of the most known coding design to solve multi-class problems is *one versus one* coding [RK04a], which consists of distinguish one class of the rest of possible classes. This kind of codification implies the use of N hypothesis to discriminate N classes. Yet using this number of hypothesis, the results obtained when dealing with high cardinality problems usually didn't reach the *random classification level*, which is known to be $1/N$ where N is the number of classes, as could be seen, this boundary shows the change between plain random classification and not random classification.

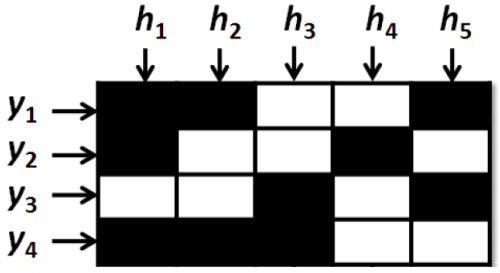


Figure 1.3: Error Correcting Output Code for a $4 - class$ problem

1.3.4 Base classification approaches

Introduction

As is shown previously, the idea of add various hypothesis or base classifiers to obtain a tool capable of solve multi-classification problems is generally extended and is one of the both most effective and robust learning strategies, as a matter of fact *Ensemble learning* is a research field within machine learning that deals with this techniques. The goal os this section is to describe the state of the art in base classifiers (also known as hypothesis) which are the mechanisms to deal with 2-class problems.

Bayes classifiers

Bayes classifiers were on the first base classifiers to appear, intending to reach the Bayes error rule. All of them are built in a probabilistic base and the distinction between the types are done based on the assumptions made when developing them.

The strongest classifier in this particular section is the Quadratic Discriminant Analysis, which can find quadratic form boundaries around the data. The development of this classifier is based on the assumption that all classes are normally distributed.

$$X \mid C_i \sim N(\mu_i, \epsilon_i) \tag{1.8}$$

Other approach of this Bayes classifiers is the Linear Discriminant Analysis, which can only find linear form boundaries on data. In this type of classifier besides the assumption that classes are normally distributed we also assume that covariance matrices are equivalent.

$$\epsilon_1 = \epsilon_2 \tag{1.9}$$

The final approach is call Naive Bayes Classifier in which besides the two assumptions made before (normally distributed data, with equivalent covariance matrices) conditional independence of the features of a sample x, given the target of the sample t_x. Which means that the probability of an observation is the product of conditional probabilities.In other words, all the features of a sample contribute independently to the probability of this sample to whether belong to a category or not. Particulary this kind of classifiers are very robust when the learning set is small because given that independence between variable is assumed, only an estimation of the variance of each category is enough and not the complete covariance matrix.

$$p(x) = p(c_i) \prod_{j=1}^{m} p(X_j = x_j \mid C_i) \tag{1.10}$$

Usually this probability based classifiers estimate the parameters of the distribution of the 2 classes by means of *Maximum Likelihood* algorithm.

Adaptative boosting and Decision Stump classifiers

Within the state of the art of base classifiers, Decision Stump classifiers combined through Adaptative Boosting (AdaBoost) are widely applied. Proving its performance on hard problems such as face detection [VJ01] or spam detection [Nic03] .

AdaBoost was formulated by Yoav Freund and Robert Schapire in 1997. This algorithm is based on the Ensemble Learning paradigm, in which a complex problem can be solved combining solutions of subproblems of less complexity than the original one.

The idea behind AdaBoost is the combination of a set of linear classifiers (also known as hypothesis). The aim of which is to generate a classifiers with a high generalization capacity. To obtain such, an iterative process is required, in which at every step weights corresponding to the performance on training data of the linear classifiers (also called weak classifiers) are updated.

The main point of this algorithm is that poorly performed classifiers at step n will increase their weight in step $n + 1$, in such a way that in iteration $n + 1$ more importance will be give to them, in order the algorithm to focus on the most complex samples to classify.

There exists various versions of AdaBoost (discrete, real, gentile, etc.) but only the way of calculating the weights of weak classifiers differ between them. Therefore we only show the main part of the algorithm.

1. Start with uniform distribution of weights $w_i = 1/N, i = 1, ..., N$.
2. Repeat for $m = 1, 2, ...M$
   - Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on training set
   - Calculate $err_m = E_w[1_{y \neq f_m(x)}], c_m = \log((1 - err_m)/err_m)$
   - Update $w_i \leftarrow w_i exp[c_m \cdot 1_{y \neq f_m(x)}], i = 1, 2, ..., N$ and normalize so that$\sum_i w_i = 1$
3. Final classifier solution $sign[\sum_{m=1}^{M} c_m \dot{f}_m(x)]$

Where weak classifiers will be $f_m(x)$ and the weights will be c_m. Generally weak classifiers are implemented by means of *Decision Stumps* which are algorithms of binary linear classification, which only have the restriction of giving a better solution that the random one.

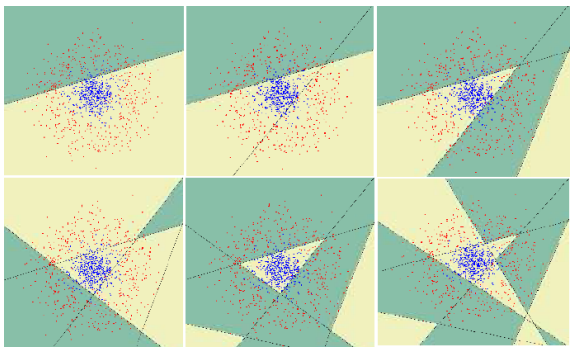


Figure 1.4: Adaptative Boosting visualization

Given its nature, AdaBoost is sensible to problems as noise in training data or outliers which are atopic values with the general distribution of the data. Which is a problem when dealing with a huge number of samples for category is arbitrarily big.

Support Vector Machine classifiers

Support Vector Machines (SVM) where first formulated in 1992 by , Boser, Guyon and Vapnik. This was a generalization of *Optimal Hyperplane* algorithm, first proposed by Vapnik in 1963. The general idea of this classifiers is to take the sample of both categories to a high dimension space, where both categories will be separated with a distance the bigger the better.

In this high dimension space the SVM will construct a linear model to distinguish both categories, this model will be the one that maximizes the distances with the closest samples of both categories, in order to obtain a high generalization capability.

Lets suppose that we have 2 categories in our training set. The aim of our SVM will be to decide to which category belongs each new point that arrives to the system. Recalling that the SVM were constructed in a high dimensional space, each point is seen as a $p - dimensional$ vector.

The aim of the algorithm is to know if the two categories of the training set can be discriminate with a $(p-1) - dimensional$ plane, commonly known as hyperplane or linear classifier. Usually a large number of hyperplanes will have the properties describe above but to select the better hyperplane we have to focus on the distance with the closest point of both categories, in other words, this distance has to be maximized in order to increment the generalization capability.

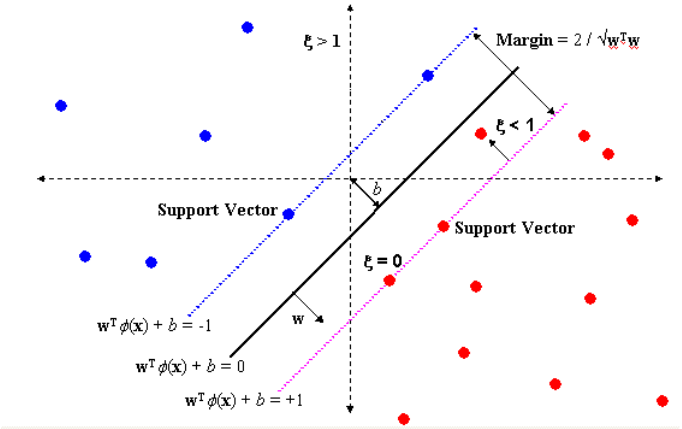


Figure 1.5: Optimal hyperplane

More formally let $S = \{(x_1, y_1), ..., (x_l, y_l)\}$ be the training set, then there exists a function of the type $g(x) = \langle w, \phi(x_i)\rangle + b$ which comes determined by a set of weights w and threshold b and there also exists a $\gamma > 0$ so that $\xi_i = (\gamma - y_i g(x_i))_+ = 0$ for $1 \leq i \leq l$. Which informally implies that the two categories of the training set are linearly separated with a margin γ.

In most of the occasions the 2-class problem won't be linearly separable in the input space. As mentioned before, the machine maps the data in a high dimension space, in other words, sees every point as $p - dimensionalvector$. To generate this high dimension spaces, *kernels* are used which well-define a scalar product. This kernel functions are common functions applied in the science fields. Between other polynomial, gaussian and sigmoidal kernels are the ones that have shown a better performance, increasing therefore the research interest on them.

Chapter 2

Methodology and strategies

The aim of this chapter is to deeply analyze, the methods and techniques used to solve a multi-class categorization problem where the cardinality of categories is arbitrarily big. As mentioned previously, the goal of the work is to describe a new approach to efficiently solve very high cardinality categorization problems, where previous approaches failed either because they try to use an unfeasible number of classifiers or because of obtained results were not statistically significant. The proposed solution stand in three main points:

1. Error Correcting Output Codes.
2. Support Vector Machine classifiers.
3. Genetic Algorithms.

Which are thoroughly analyzed in the following pages.

2.1 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.

2.1.1 Motivation

High cardinality problem suggest a huge quantity of data with probably very complex distributions. In [Pol06] approaches based in Ensemble Learning are suggested for this kind of frameworks. Given the Ensemble Learning approach accomplish a set of features which very useful when dealing with great quantity of data. There exists various reasons to use an ensemble based system.

Statistical reasons

Readers familiarized with neural networks or other machine learning systems are very aware that having good performance on the training set doesn't mean to have a good generalization capability. A set of classifier with good performance over the training set may have distinct generalization capability. Even a set of classifiers with a good generalization capability over the training set may perform (and usually do) bad in the field, having its poorest performance if the training set is not representative enough of the real distribution of the data. In those cases, combining the output of classifiers in a smart way could decrease the odds of choosing a particularly very poor generalization capability classifier.

Huge quantity of data

In some cases the volume of data to process is simply very high to be treated by an only classifier. In those cases, partitioning the dataset in different subsets training for each of them a classifier and combining the outputs tends to be much more efficient.

Divide and conquer

Beside of the great quantity of data, some problems are just to complex to be treated by an only classifier. Particularly, the boundary that discriminates 2 categories may be extremely difficult to approximate, or lay outside the scope of the classifier.

Let's assume to have access to a base classifier that can find elliptic or circular boundaries in the data. Such classifier will not learn the boundary that is shown in section (a) of the image 2.1. Let's consider then, a boundary generated by a set of classifiers pertaining to an ensemble such as the one in section (b). A final prediction based in voting by an enough number of classifiers may learn such a complex boundary as shown.

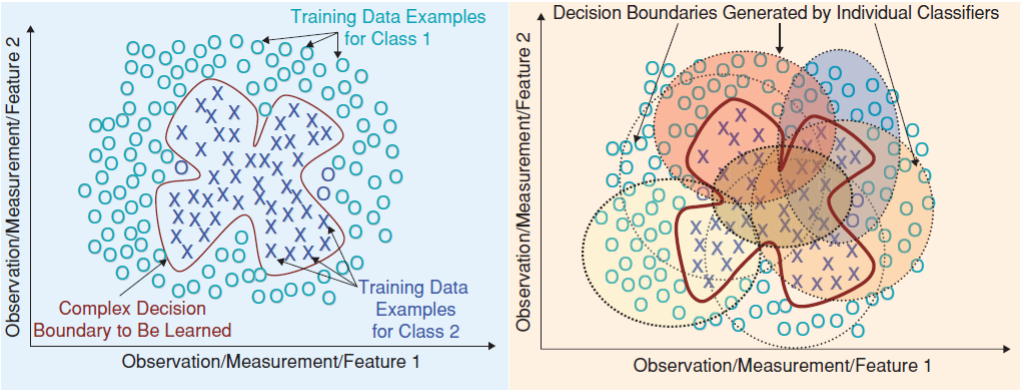


Figure 2.1: Divide and Conquer strategy : real boundary (a), boundary approximated by the ensemble(b)

Mainly, the system suggest a "divide and conquer" approach where instead of learning the whole dataset, subset of lower complexity are treated. In this sense, the complex original boundary can be approximated by a set of classifiers smartly combined.

2.1.2 Error-Correcting Output Codes

Introduction

Error Correcting Output Codes, previously introduced, are an ensemble based method which deals with multi-class categorization problems from a "divide and conquer" point of view. As previously mentioned, multi-class categorization problems have been widely studied. Nevertheless, one of the main approaches in dealing efficiently with those problems was the ensemble approach. In these sense, Error Correcting Output Codes have been broadly applied in state of the art problems, such as facial verification [KGWM01].

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 2.2 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:

$$h_1(x) = \begin{cases} 1 & \text{if } x \in \{c_3\} \\ -1 & \text{if } x \in \{c_1, c_2, c_4\} \end{cases} \quad , \ldots , \quad h_5(x) = \begin{cases} 1 & \text{if } x \in \{c_2, c_4\} \\ -1 & \text{if } x \in \{c_1, c_3\} \end{cases} \quad (2.1)$$

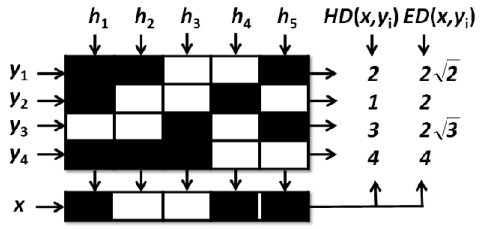


Figure 2.2: Binary ECOC design for a 4-class problem. An input test codeword x is classified by class c_2 using the Hamming or the Euclidean Decoding.

ECOC coding

The coding step of the ECOC framework is the step of finding the matrix M which defines the codewords of every category in the problem. There are different strategies depending on the number of symbols in the matrix.

The standard binary coding designs are the one-versus-all [PGCP65] strategy with N dichotomizers and the dense random strategy [ASS02], 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 [TR98] and the sparse random strategy [ASS02], with $\frac{N(N-1)}{2}$ and $15 \log_2 N$ binary problems, respectively.

The one-versus-all design is the most used. Consisting of distinguishing each category from the rest. Given N classes, this design establish a number of N classifiers, in other words, it has a code length of N bits. As can be seen this is not a recommended design to deal with high cardinality problems provided that the number of classifiers to train is the same as the number of classes.

On the other hand, there exists random based strategies, such as, "dense random", in which a high number of coding matrices of length n are generated, where values $-1, +1$ are equiprobable. Works on the performances of such matrices have suggested a codeword length of $n = 10 \lg(N)$, where N is the number of classes. Therefore by means of this technique a improvement on computational cost is accomplished. The optimal matrix of the set is the one that maximizes the Hamming distances, baring in mind that both values $\{+1, -1\}$ have to appear in each column.ç

Another coding design is known as one-versus-one. In such, a third symbol is incorporated in the design of the M, the zero symbol, which implies that the

categories affected by this symbol are not taken into account by the base classifier. Nevertheless, independently of the performance, the use of three symbol in this particular design implies the increment of the length of the codeword. As a matter of fact, a trade-off between strategies may perform better that each of them separately.

ECOC decoding

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 2.2, the new code x is compared to the class codewords $\{y_1, ..., y_4\}$ using Hamming [PGCP65] and Euclidean Decoding [ASS02]. 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 [EPP09]: those based on similarity measurements, including the Hamming and Euclidean decoding [PGCP65], probabilistic approaches [PPF04], and loss-functions strategies [ASS02].

- **Hamming Decoding (HD)**: in this kind of decoding the Hamming distance between the predicted word of the ECOC classifiers and the distinct matrix rows M id calculated. The final prediction is the one that minimizes the distance.

$$HD(x, y_i) = \sum_{j=1}^{n} (1 - sign(x^j \cdot y_i^j))/2 \tag{2.2}$$

- **Euclidean Decoding (ED)**: this metric is based on calculating the Euclidean distance between the ECOC classifiers response and each codeword of the matrix M. Finally the prediction is the one that minimizes the distance.

$$ED(x, y_i) = \sqrt{\sum_{j=1}^{n} (x^j - y_i^j)^2} \tag{2.3}$$

- **Loss-based Decoding (LBD)**: This is the decoding design implemented in the system, therefore is deeply analyzed. The loss-based decoding strategy chooses the label l_i that is most consistent with the predictions f (where f is a real-valued function $f : \rho \to R$, in the sense that, if the data sample ρ was labeled l_i, the total loss on example (ρ, l_i) would be minimized over choices of $l_i \in l$ where l is the complete set of labels.

$$LB(\rho, y_i) = \sum_{j=1}^{n} L(y_i^j \cdot f^j(\rho)) \tag{2.4}$$

Where L is a loss function that depends on the nature of the binary classifier. The one used in the implementation of the system is $L(\theta) = -\theta$. The final decision is achieved by assigning a label to example ρ according to the class c_i that obtains the minimum score.

As can be seen is next sections the new ECOC coding design implies often the use of a third symbol, the zero symbol, which implies that certain classes are not taking into account by the base classifier. Therefore, a ternary ECOC decoding design has to be integrated. As said before the LBD decoding provide us with a clear framework for ternary decoding design.

2.1.3 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 (see Figure 2.3). 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 [RK04b]. Recently, this codeword length has been reduced to $N-1$ in the DECOC approach of [PRV06], 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 [UW04, CS02, PRV06, EPP10]. 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. Figure 2.3 shows the number of dichotomizers required for the ECOC configurations of the state-of-the-art for different number of classes. The considered codings are: one-versus-all [PGCP65], one-versus-one, Dense random, Sparse random, DECOC, Forest-ECOC, Sub-class, and ECOC-ONE [TR98, ASS02, PRV06, EPP10].

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:

$$B = \lceil \log_2 N \rceil \tag{2.5}$$

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

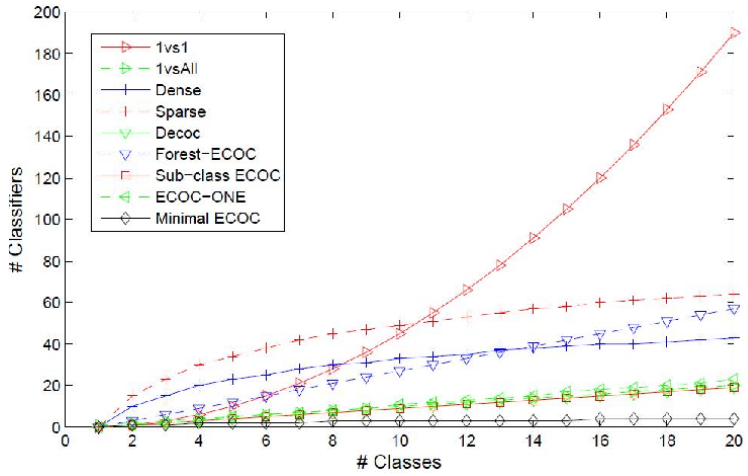


Figure 2.3: Minimum number of dichotomizers required for each ECOC configuration and different number of classes.

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 2.4. The white and black positions correspond to the symbols +1 and -1, respectively. The reduced number of classifiers required by this design in comparison with the state-of-the-art approaches is shown in the graphic of Figure 2.3.

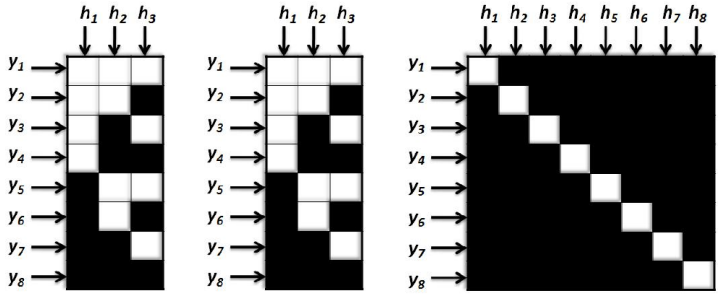


Figure 2.4: (a) Binary Minimal, (b) Gray Minimal, and (c) one-versus-all ECOC coding designs of a 8-class problem.

The algorithm for Minimal ECOC coding is the following:

Data: Number of categories of the problem N
Result: Minimal ECOC codeword matrix M
Generate a random permutation of numbers from 1 to N;
while *cannot split in iteration* i *the partitions generated in iteration* $i-1$ **do**
  split the permutation/s breaking in the central point;
end
Codify the $i-th$ level of the tree-structure as the $i-th$ column of the M matrix where left sons will have value 1 and right sons the value -1;

Algorithm 1: Minimal ECOC coding

Besides exploring predefined binary or Gray minimal coding matrices, we also propose the design of a different minimal codification of M 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 a $N-$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.

2.1.4 Incremental ECOC Coding

Introduction

In the process of defining the Minimal ECOC coding, the issue of performance is always present. It's easily understandable that the definition of such a few number of classifiers per problem (sub-linear with the number of classes) can lead to an under-fitting situation. In addition, when dealing with multi-class categorization problems it's completely understandable that treating this huge number of categories with such a few number of classifiers may affect the performance of the system.

Consider the following scenario in image 2.5 where the classes *star* and *circle* are very confused in the left image, this confusion made by one of the minimal ECOC classifiers, could lead to a lost of generalization on the whole system. In the Minimal ECOC coding design a solution to this problem is introduced, incrementing the ECOC matrix M by adding a base classifier that would be focused on the problem that produced the greatest lost of generalization performance. Therefore, some errors made by the original ECOC ensemble are repaired.

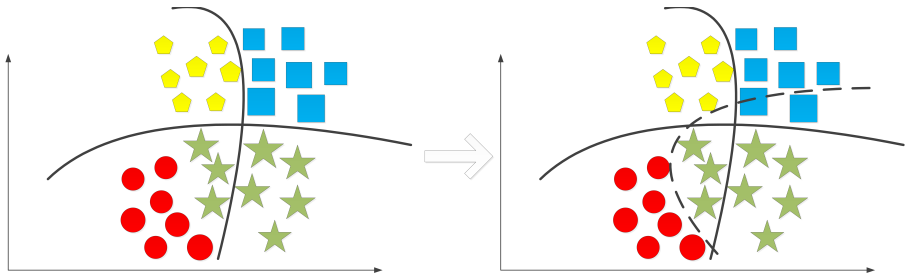


Figure 2.5: ECOC incremental step

Incrementing the ECOC matrix

Let's assume that in some step of our genetic optimization an ECOC matrix is evaluated, therefore, we can obtain the *confusion matrix* $Conf \in [0, +\inf)^{N \times N}$ for it. In this matrix we can see the real target value and the predicted target value for every sample classified. When a good classification is done all values of this matrix are contain in the diagonal, thus they're correctly classified. Using the properties of this matrix we can find the category which has less correctly classified instances (it's the most confused category) and find with which category this confusion is maximized. As a matter of fact, the confusion calculated Cf is weighted by the number of examples on each category.

$$Cf = \frac{\sum Conf_{\forall j,i}}{\sum Conf_{\forall j,\forall i}} \times \frac{(1 - Conf_{i,i})}{\sum Conf_{\forall j,i}} \tag{2.6}$$

In this sense, at every evaluation a ECOC matrix in the evolutionary step, the more complex problem (the one that generates maximum lost of performance) is solve by adding a new base classifier focused on the categories that maximize the confusion between them, as can be seen in the image 2.6.

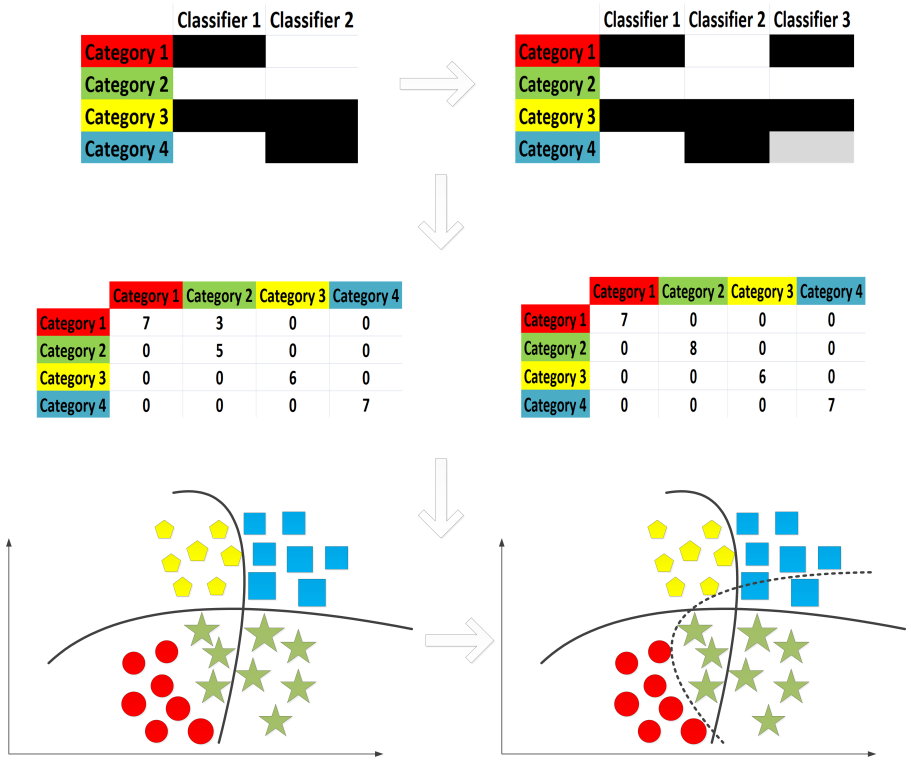


Figure 2.6: Confusion-based solver

This incremental step does not increment the ECOC matrix until infinitum, it's constrained not to increment the number of base classifiers more than twice the length of the original Minimal ECOC coding number of classifiers. In addition, when a confusion is going to be solved, the new base classifier's aim is to discriminate between the most confused classes. But if in a previous step this problem was treated, but in fact there was no correction of confusion the new classifier will treat the same problem. In order not to introduce two equivalent columns in the ECOC matrix M, the values not concerned with most confused classes are randomly changed to $0, 1, -1$ with equal probability.

Finally a parameter can adjust how much has the overall performance to increment in order to include the new classifier.

Confusion solvers

When solving the confusion between two classes, various alternative are analyzed. Recalling the figure 2.5, is obvious that a wide set of classifiers could be added to the ECOC matrix M in order to solve the maximum confusion problem. Taking into account the global complexity of the multi-class categori-

zation problem, two approaches are developed.

- **One-vs-one solver**: In this kind of coding, a new column is added to the ECOC matrix M. This new base classifier only takes into account the discrimination of the two categories that maximize the confusion between them. An example can be seen in figure 2.5.
- **Confusion-based solver**: In the second approach not only the two classes that maximize confusion are included in the coding, but also, more categories are taken into account. The general idea is to improve the performance of the incremental codding by supporting the two more confused classes with the ones that minimize the confusion between them. An example is shown in figure 2.7. A parameter can control the percentage of the total information that the classifier can include. In this sense, we can control the number of categories that support each one of the two most confused classes in order to increment the performance of the base classifier when solving the confusion.

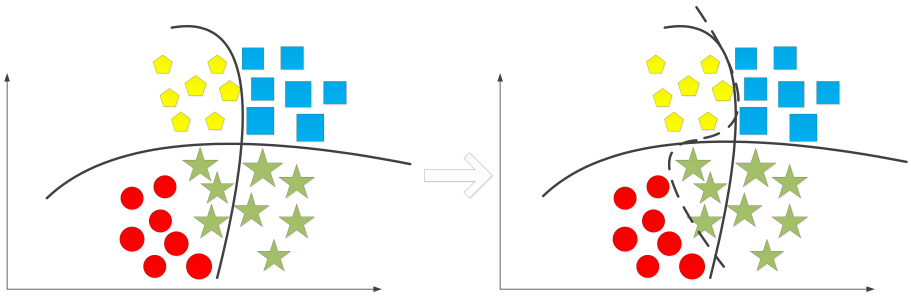


Figure 2.7: Confusion-based solver

2.2 Base classifier: Support Vector Machines with RBF kernel

2.2.1 Introduction

Support Vector Machines (SVM) were formulated in 1992, by Boser, Guyon and Vapnik, which was a generalization of the *optimal hyperplane* algorithm first proposed by Vapnik in 1963. The general idea of this classifiers is taking the samples of the training set to a high dimension space in where those categories are separated for a distances the larger the better. In this high dimension space the SVM will construct a model to discriminate those classes, this model will maximize the distance with the closest sample of both categories in order to maximize the generalization capability. Such classifiers have been applied over a wide set of environments showing great performance results. For example,

[BZ09] where the authors use SVM to predict the position of subcellular proteins. Another interesting work is the application to image recognition [ACSS02].

2.2.2 Motivation

Large-scale problems and the minimal ECOC coding design proposed, suggest the use of a very compact codeword, in other words, the ensemble of base classifiers is very reduced. This reduction of the number of classifiers can be done assuming the use of very robust classifiers, showing the higher performance in the state of the art, provided that the ECOC coding design chosen is not able to correct many errors.

Besides, SVM have an explicit dependency on the data (by mean of the support vectors), in such a way that the model is easily interpretable, which is a great point when dealing with great quantities of data.On the other hand, the learning task of a SVM implies the optimization of a convex function, free of relative minima and in which only exists a absolute minima.

The last point of using SVM is that few parameters have to be tuned, normally the soft margin constraint and the kernel parameter.

2.2.3 Methodology

Let's suppose that we have two categories in our training set. The aim of our classifier will be to decide in which category is classified every new sample. In this sense, SVM see each sample as a $p-dimensional$ vector.

The goal of the algorithm is to find out wether the two categories in the training set can be discriminated with a $(p-1)-dimensional$ plane, commonly known as an hyperplane or linear classifier. Generally, there will exist a arbitrarily huge number of such linear classifiers, therefore another constraint is necessary to filter the set of classifier an obtaining the best one. This criteria is that the best classifier has to maximize the distances between itself and the closest points of each category, in other words, maximize the margin between the hyperplane an the categories.

Linear cases in input space: Optimal Hyperplane

Let's assume that the two categories are perfectly separable in the input space. In other words, the can be separated by means of a linear classifier. Let 2.7 be the dataset and x_i y $y_i \in \{-1, +1\}$ their labels, then exists a linear function 2.8 weighted by w. In this sense, our decision boundary 2.9, keeps invariant to both a normalization of both w_i and b 2.10.

$$S = \{(x_1, y_1), ..., (x_n, y_n)\} \tag{2.7}$$

$$g(x) = \langle w \cdot x_i \rangle + b \tag{2.8}$$

$$D(x_i) = sign(\langle w \cdot x_i \rangle + b) \tag{2.9}$$

$$w \rightarrow \lambda w, b \rightarrow \lambda b \tag{2.10}$$

Once the decision boundary is obtained, the following step is maximize the margin γ that is between the hyperplane and the closest point of each category. Geometrically speaking, the margin is defined by the projection of the vector $x_1 - x_2$ (where x_1 is a support vector belonging to $+1$ category and x_2 is a support vector the belongs to the class -1) on the normal of the linear classifier, as we can see in the image 2.8

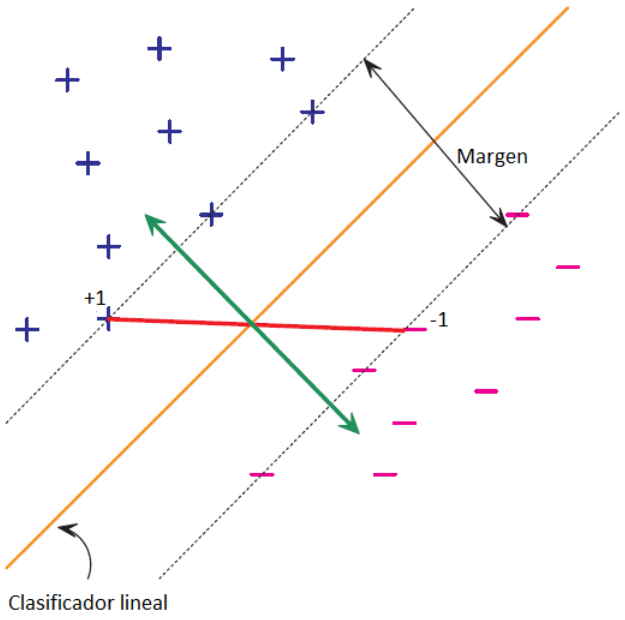


Figure 2.8: Margin of a Support Vector Machine

Formally the margin can be defined as:

$$\gamma = 1/\|W\|_2 \tag{2.11}$$

In this sense, the task of maximizing the margin formulated in 2.11 is equivalent to minimize 2.12 constrained by 2.13.

$$\phi(w) = \frac{1}{2}(w \cdot w) \tag{2.12}$$

$$y_i[(w \cdot x_i) + b] \geq 1 \tag{2.13}$$

In other word, the problem is translated in finding an optima of the cost function 2.14, where the $alpha_i$ are the Lagrange multipliers.

$$L(w,b) = \frac{1}{2}(w \cdot w) + \sum_{i=1}^{m}(\alpha_i[y_i((w \cdot x_i) + b) - 1]) \tag{2.14}$$

One of the most interesting aspects in the optimization of this functional, is that the data (x_i vectors) appear in a scalar product with the weights. Therefore, the optimization process of such functional is independent of the dimensionality of the data.

Usually is very difficult to find situations where the data is linearly separable, for instance in out system, every classifier of the ensemble will take into account 2 wide sets of categories to discriminate, so we can assume that the distribution will be very complex. Here's where lays the power of SVM, because in situations where a linear classification is unappropriated in the input space, SVM can map the data in a higher dimension space, where as a matter of fact the data can be linearly separable.

Non linear cases in input space: high dimension space mapping

As explained previously, in most of the cases the data will not be linearly separable in the input space. By means of the property seen in 2.14 SVM can map this data which are not linearly separable, into high dimension spaces (also known as feature spaces) where the situation changes and a hyperplane can easily discriminate these data.

Formally, to obtain a better representation of the data where is not linearly separable in the input space, we can project them in a high dimension space by means of 2.15. In other word, we use a map of 2.16 type.

$$x_i \cdot y_j = \phi(x_i) \cdot \phi(y_j) \tag{2.15}$$

$$x_i \rightarrow \phi(x_i) \tag{2.16}$$

The only constraint to perform the mapping, is that the feature space where data is projected has to be an Hilbert space (although in some cases a pre-Hilbert space could be enough). Such space is a $n-dimensional$ generalization of an Euclidean space. In which a dot product is well defined.

In this situations, the mapping function $\phi(x_i) \cdot \phi(x_j)$ is known as a kernel. Therefore we can define a kernel as the dot product of two points projected in the feature space 2.17.

$$K(x_i, y_j) = \phi(x_i) \cdot \phi(y_j) \tag{2.17}$$

In this sense, we don't need to known the shape of the data mapping in the feature space, because it would be implicitly defined in the calculus of the dot product in such space.

There exists a variety of kernels but because of the scope of this study we are only going to introduce the polynomial and the RBF Gaussian kernel, which is deeply analyzed in the following sections.

- **Polynomial kernel**: The polynomial kernel is one of the most used to model a non linear problem. This kernel is based on constructing a polynomial function of degree d (where d is the dimension of the feature space) which accomplishes an acceptable classification of the input data. In 2.9 it can be seen how this kernel establishes a classification much more acceptable that any other linear classifier by its own. Nevertheless, in the image it can be seen that this is not the best classification possible, it can be improved.

$$K(x, x') = (\langle x, x' \rangle + 1)^d \tag{2.18}$$

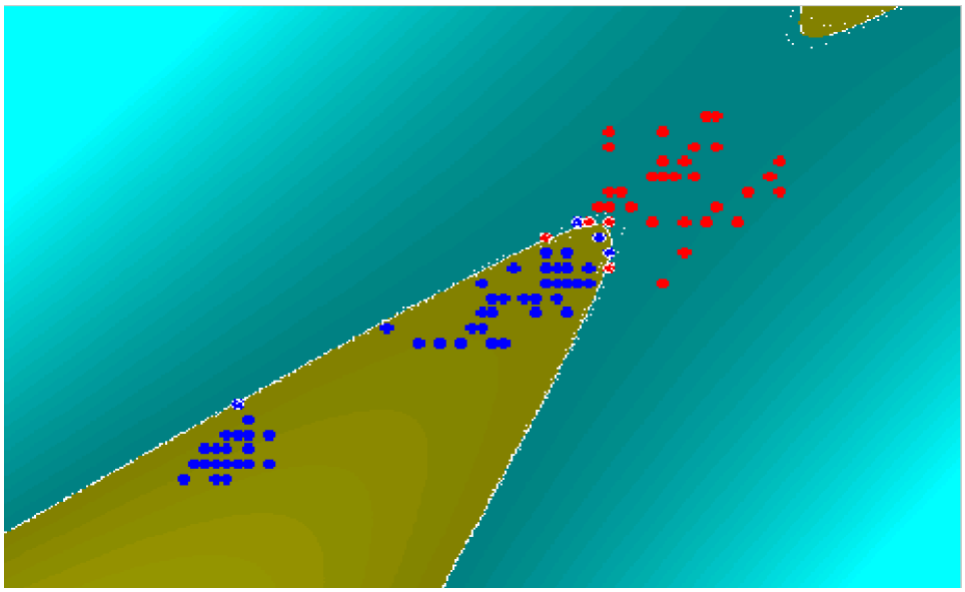


Figure 2.9: Decision boundary generated by a polynomial kernel

- **Kernel RBF**: Another type of kernels are the Radial Basis Function kernels (RBF). This kind of kernels have accomplish an special attention providing their high performance even in very complex situations, as for instance, the one regarding this work. We focus on the Gaussian case, where is required the fitting of an extra parameter, the γ parameter. It can be seen in 2.10 that for the distribution of the data, the RBF kernel has a similar or slightly better performance that a the polynomial kernel, but generally in literature, the performance of the RBF is observed to be better than a polynomial kernel.

$$K(x, x') = \exp(\frac{-\|x - x'\|^2}{2\sigma^2}) \tag{2.19}$$

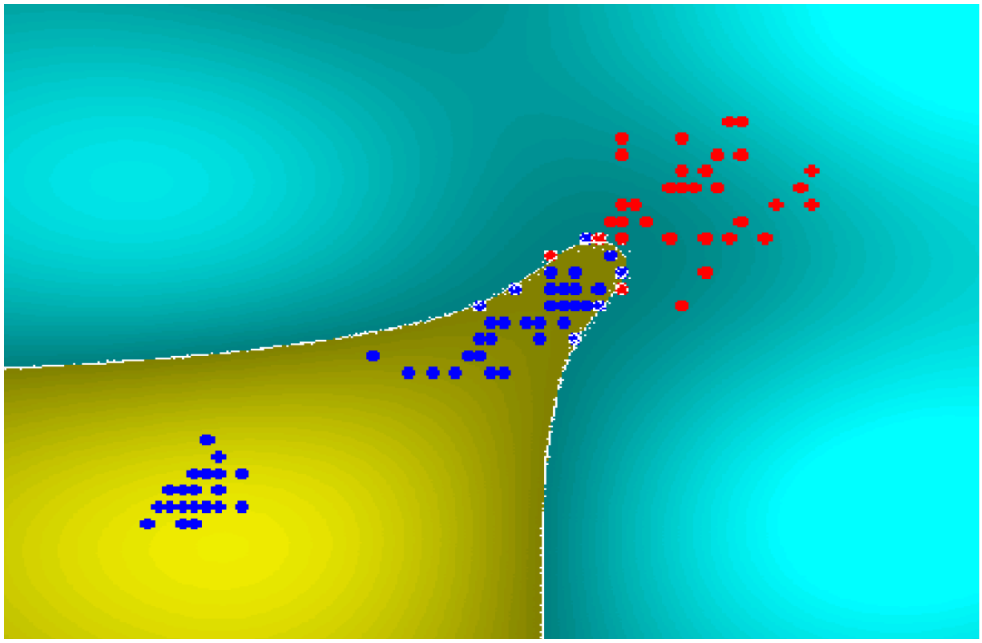


Figure 2.10: Decision boundary generated by a RBF Gaussian kernel

Mathematically, for a kernel to be valid it has to accomplish the Mercer condition, formally let $K(x, x')$ be a kernel, it will be valid if:

$$\int K(x, x')g(x)g(x')\ dxdx' \geq 0 \tag{2.20}$$

In other words, the kernel has to be Positive Semi-defined (PSD) 2.21 and 2.22, then there exists a function 2.23 which generates a Hilbert space with well defined dot product.

$$\sum_{i,j} K(x_i, x_j)c_i c_j \geq 0 \tag{2.21}$$

$$\{c_1, ..., c_n\} \ \in \ \mathbb{R} \tag{2.22}$$

$$K(x, y) = \phi(x) \cdot \phi(y) \tag{2.23}$$

There exists some kernels, such as sigmoidals, which violate the Mercer condition. There are considerated out of scope and therefore are not introduced.

Soft margin hyperplane

One of the most common problems to solve in machine learning task is overfitting. This situation takes place where the system is trained very hard over a

training set in such a way that it learns the idiosyncrasies of the distribution. As we can seen in the image 2.11, where the black boundary represents a good generalization measure, whereas the green boundary fits to much the outliers of the data and therefore its generalization capability is diminishes.

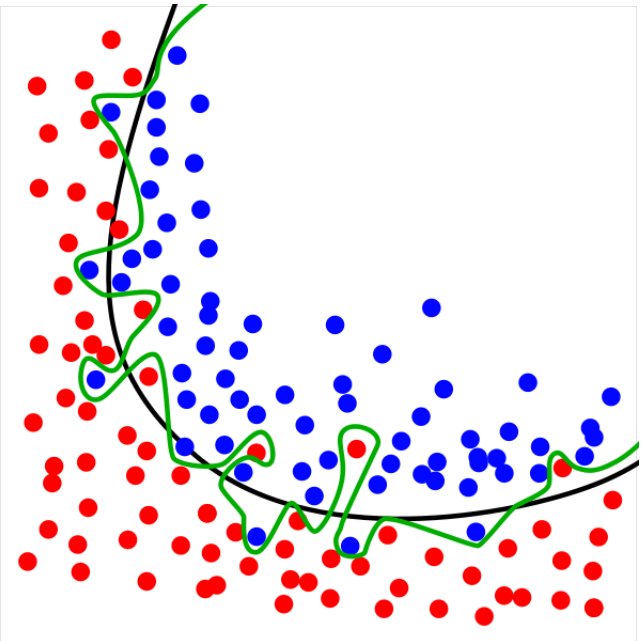


Figure 2.11: Overtraining situation

Overfitting implies a great lost of generalization capability of the classifiers, driving the system to poorly performance situations. Idiosyncrasies usually are outliers that appear due to the noise in the data.

The strategy to deal with this problem in SVM is the introduction of a soft margin, which can be modified to minimize the classification error, or in other words, the overfitting to the training data.

Formally, we want to upper-bound the value of the Lagrange multipliers (the value of which can be seen as the importance they have for the construction of the hyperplane) α_i by a value C, as in 2.24. To do so, a positive slack variable ξ_i is introduced obtaining 2.25, having to reformulate the cost function to reach 2.26 where $\alpha_i \geq 0$ and $r_i \geq 0$ are the Lagrange multipliers.

$$0 \geq \alpha_i \geq C \tag{2.24}$$

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \tag{2.25}$$

$$L(w, b, \alpha, \xi) = \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i [y_i((w \cdot x_i) + b) - 1 + \xi_i]) - \sum_{i=1}^{m} \xi_i r_i \tag{2.26}$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} \alpha_i x_i y_i = 0 \tag{2.27}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{2.28}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - r_i = 0 \tag{2.29}$$

Taking partial derivatives of the new cost function 2.29 we can obtain 2.24. Therefore we obtained a hyperplane in which the maximum importance that some points will have over it can be bound and as a direct implication overfitting is controllable.

SVM with a Gaussian RBF kernel

In literature, SVM with a RBF kernel have obtained very good results. Particularly, Gaussian RBF kernels have reached the best performances in base classification tasks. This is not a coincidence since, this types of function have been widely applied in ANN for a long time.

A Radial Basis Function or RBF, is function of real valued which valued only depend on the distance of the origin $\phi(x) = \phi(\|x\|)$ or alternatively from a source point c called center, $\phi(x, c) = \phi(\|x - c\|)$. Any function $\phi(x)$ which satisfies the property $\phi(x) = \phi(\|x\|)$ is a RBF function. The metric for the calculus of the distance is usually the Euclidean, although other metrics exists. The most robust RBF function in SVM is the Gaussian, formulated in 2.30.

$$\varphi(r) = \exp(-\beta r^2) \; para \; \beta > 0 \tag{2.30}$$

2.3 Evolutionary optimization

2.3.1 Introduction

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, which means that we can construct $V_{2^B}^N = \frac{2^B!}{(2^B-N)!}$ different ECOC matrices. Nevertheless, in the ECOC framework, one matrix

and its opposite (swapping all zeros by ones and viceversa) are considered as the same matrix, since both represent the same partitions of the data. Therefore, the approximated number of possible ECOC matrices with the minimum number of classifiers is $\sharp M = \frac{V_{2^B}^N}{2} = \frac{2^B!}{2(2^B-N)!}$. In addition to the huge cardinality, it is easy to show that this space is non contiguous, because a change in just one bit of the matrix may produce a wrong coding design.

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 use to be formulated as optimization processes where the solution space is not differentiable or is not well defined. On 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 [BC95], a new family of evolutionary methods is striving to find a place in this field. In contrast with 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.

2.3.2 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* [Hol75]. When the solutions or individuals are transformed in order to be represented in a chromosome, the original values (the individuals) are referred as *phenotypes*, and each one of the possible settings for a phenotype is the *allele*. In this sense, an ECOC is defined as a structure with different parameters (coding matrix,performance of each classifier, parameters of base classifiers and confusion matrix). In next sections, we explain how this individual can perform on crossover and mutation operations specifically defined for them.

2.3.3 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 better 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 the $\mathbf{w}^*$ 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 over a subset of the samples, while the rest of the samples are reserved for a validation set, and the adaptation value ξ is the classification error over that validation subset. The adaptation value for an individual represented by a certain chromosome ζ_i can be formulated as:

$$\varepsilon_i(X, Y, M_i) = \sum_j \delta_j(x_j, M_i \neq y_j) \tag{2.31}$$

where M_i is the ECOC matrix encoded in ζ_i, $X = \langle x_1, \ldots, x_N \rangle$ 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.

2.3.4 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, a new crossover operation designed specifically for ECOC's is used in order for each generation to perform better that the previous. In order to introduce variations to the individuals, we use mutation is also designed specifically for this type of individuals. 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 PBIL 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.

2.3.5 ECOC-specific crossover operator

Motivation

When defining ECOC individuals as explained in previous sections one must take into account the possible lost of improvement due to the use of general crossover and mutation operators. In this sense, if the crossover operator is not specifically design or adapted to the type of population it can generate new individuals that in best cases won't perform as well as any of it's parents and in the worst cases the will generate individuals that may not respect the specific non-avoidable characteristics of the individuals.

In the ECOC framework this can be explained very clearly. For such explanation, first, we have to take into account the main property of the ECOC coding matrix M.

1. Every codeword, that is, every row of the matrix, has to be different.

Let's now consider a single point crossover operator. Let's now consider a codification in which the chromosomes are the columns of the M matrix. Now let's consider two parents P_1 and P_2 then by means of the single point operator, both parents are divided by a random point and each one of the chains generated is appended to the complementary chain of the other parent. It can be seen that this type of crossover operator in most cases will generate individuals that not maintain the specific ECOC property.

Crossover design

For the purpose of this thesis a new crossover operator specific for ECOC individuals are designed. The aim of a crossover operator is to combine the best parts of each individual in order to generate an individual that combines the best parts of his parents and it may perform better than both of them.

In this sense, we have to take into account that the better the performance of each base classifier of the ensemble the better the performance of the ECOC. So what we are seeking for is the best classifiers of each parents that can be combined in it's son. Before introducing the crossover algorithm we recall the components involved in the crossover operator of the ECOC individual consists of:

- The coding matrix M
- The performance of each classifier $Perf$

The algorithm is the following:

Data: Two ECOC individuals $ECOC^1$,$ECOC^2$ sorted by the performance of the best classifier on each ensemble
Result: $ECOC^{out}$

$M^{out}_{i,1} \leftarrow M^{ECOC^1}_{i,\max(Perf^{ECOC^1},1)}$;
$FollowingParent = ECOC^2$;
$repetitionsAllowed \leftarrow ceil(NClasses/2)$;
for $k \leftarrow$ *Classifiers in* $ECOC^1$ -1 **do**
  $j \leftarrow 1$;
  $rep \leftarrow NumberOfRepetitions(M^{out}_{i,1}, M^{FollowingParent}_{i,\max(Perf^{FollowingParent},j)})$;
  while $rep > repetitionsAllowed$ *and* $j <$ *Classifiers in* $FollowingParent$ **do**
    $j \leftarrow j + 1$;
    $rep \leftarrow NumberOfRepetitions(M^{out}_{i,1}, M^{FollowingParent}_{i,\max(Perf^{FollowingParent},j)})$;
    if $rep < ceil(NumberOfClasses/2)$ **then**
      $found \leftarrow true$;
      $break$;
    end
  end
  if $found$ **then**
    $M^{out}_{i,k+1} \leftarrow M^{FollowingParent}_{i,\max(Perf^{FollowingParent},j)}$;
    $Perf^{out}_{k+1} \leftarrow \max(Perf^{FollowingParent}, j)$;
  else
    $M^{out}_{i,k+1} \leftarrow generateValidColumn(M^{out}, ceil(rep))$;
    $Perf^{out}_{k+1} \leftarrow 0$;
  end
  if $FollowingParent == ECOC^2$ **then**
    $FollowingParent = ECOC^1$;
  else
    $FollowingParent == ECOC^2$;
  end
  $repetitionsAllowed \leftarrow ceil(repetitionsAllowed/2)$;
end

Algorithm 2: ECOC-specific crossover algorithm

Crossover explanation

In this section both a graphical and analytical analysis of the ECOC-specific crossover operator will be performed. As said in previous sections the aim of the crossover function is to find individuals more adapted to the environment (in our case those who have a greater performance) than their parents.

In the ECOC framework will perform as better as the performance of each base classifier, in other words, if the performance of the base classifiers improve,

the overall performance of the ECOC will also improve. Therefore, to make an ECOC individual to improve we have to provide it with the best performing classifiers of his parents, but we have to take into account the restriction of the ECOC framework exposed in 2.3.5. Taking this into account, the newly designed algorithm look first looks for the best classifier of both parents and then seeks for the best classifier of the complimentary father that can assure that the ECOC property will be maintained.

Let's consider the far known example of previous sections.

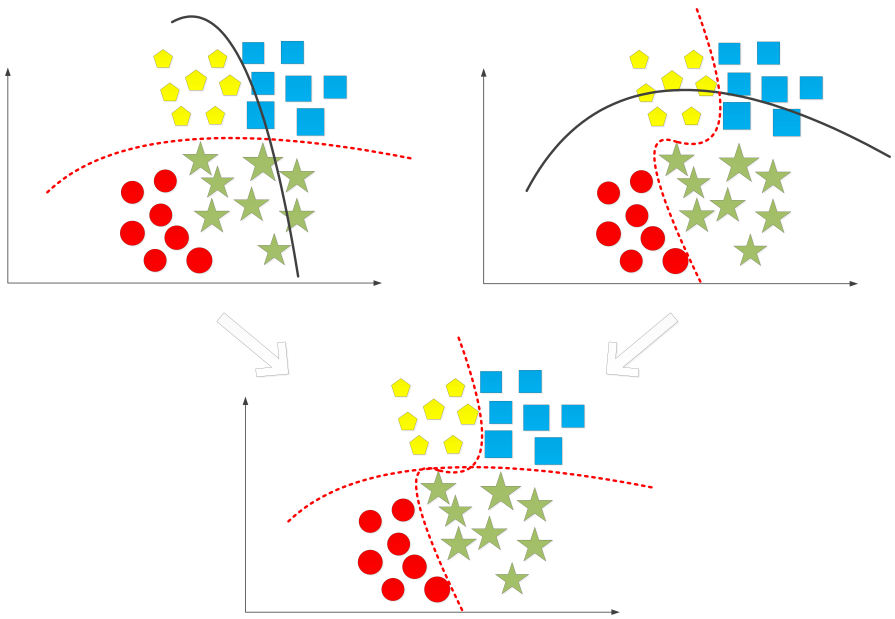


Figure 2.12: Crossover operator in a 4-class toy problem

In the image 2.12 we can see two graphical representation of a 4−class problem in which only 2 classifiers are needed by the coding design in order to solve the problem for each one of the solutions. Let's now consider the ECOC-specific crossover operator in this scenario. As wee can see the classifiers that show a greater performance are highlighted in red and dashed lines. This classifiers would be the best candidates for building a new ECOC individual. Since the ensemble produced by them does not break the ECOC property 2.3.5 (they univocally distinguish every class) they will form a ECOC offspring which performs better than any of their parents.

Let's now considerer a harder scenario, with a 5-class problem.

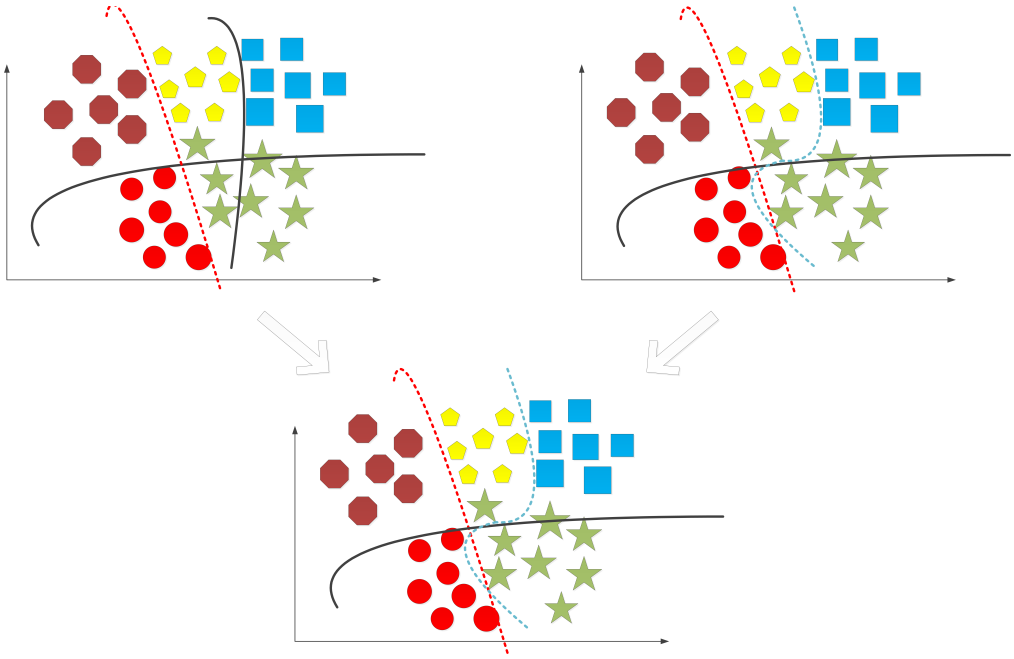


Figure 2.13: Crossover operator in a 5-class toy problem

The figure 2.13 shows a harder scenario, in which, we will follow the crossover operator flow. First we choose the best classifier of both parents, let it be the red dashed line at the top left situation. Then we seek in the other parent (top right situation) for a classifier that keeps the ECOC property starting by the best classifier (the red dashed line) but as we can see this classifier doesn't hold the ECOC property, therefore is not included. Then we look for the second best classifier (the light blue dashed line) and as a matter of fact hold the property, therefore is included. Finally we return to the first parent and choose the final classifier that keeps the property.

In both situations, we have assumed that there was always a classifier in any of ECOC parents that hold the ECOC property. There could be the case that there's no such classifier, then the crossover operator is allowed to generate a new classifier to make the ECOC hold its property and distinguish all the classes univocally.

2.3.6 ECOC-specific mutation operator

Motivation

As explained previously, the crossover and mutation operators have to be specifically defined if the individuals that are codified have some type of properties or constraints that must always hold. To define the mutation operator we have

to take into account that a mutation is a random operation in which the individual is not assure to be better adapted to the environment after the mutation. Taking this into account an ECOC-specific mutation is operator by swapping two random rows of the ECOC coding matrix M.

Mutation design

The ECOC-specific mutation algorithm is the following. In this operator, two random rows are swapped.

Data: A ECOC individual $ECOC^{mut}$
Result: The mutated ECOC, $ECOC^{out}$

$p \leftarrow rand$(1,Number of Classes);
$k \leftarrow rand$(1,Number of Classes);
$M_{k,j}^{out} = M_{p,j}^{mut}$;
$M_{p,j}^{out} = M_{k,j}^{mut}$;

Algorithm 3: ECOC-specific mutation algorithm

2.3.7 Training the base classifiers

In [RK04b], 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 [LdC08], where individuals correspond to a pairs of genes, and each gene corresponds to the binary codification of a floating point value.

Figure 2.14 illustrates an iteration of the evolutionary Minimal ECOC parametrization for a toy problem of three face categories. Given an input chromosome ζ (up) that codifies a valid minimal matrix M, for each dichotomizer, an evolutionary approximation of the classifier parameters (C, γ) is performed. From left to right of the image, we show the Minimal ECOC matrix codifying the 3-class problem, the feature spaces, and the search space of (C, γ). Each decision boundary shows a possible certain solution given by the GA. In this sense, the GA starts giving solutions (pairs C, γ) that do not fit a robust de-

cision boundary at the beginning, but after convergence, the GA increases the adjustment of these boundaries. The final parameters are those that minimize the error over a validation subset. The evolution of the best individual is shown in the toy error surface on the right of the image. Finally, the evolution of this matrix returns the adaptation value ξ of the current individual (current minimal configuration), estimated as the minimum classification error computed.

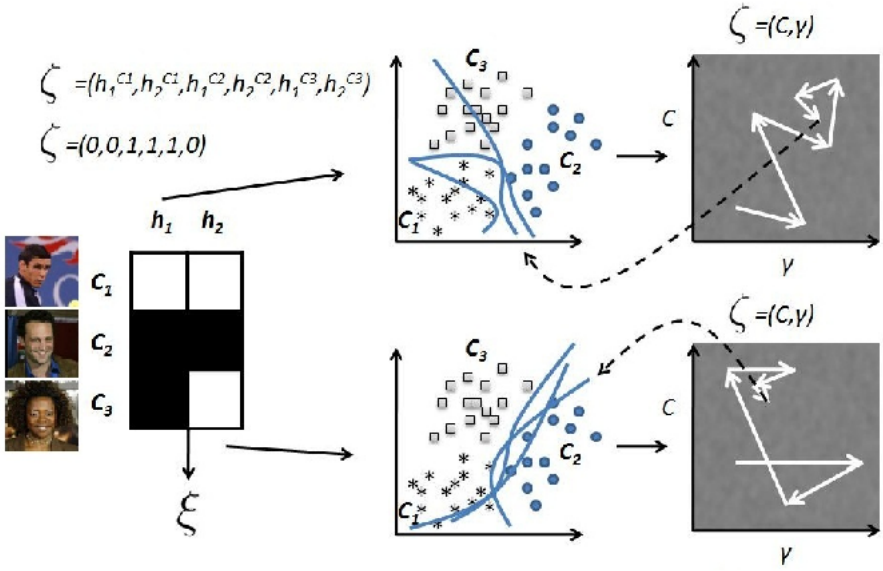


Figure 2.14: Evolutionary optimization for toy problem.

In order to save time, a historical of column codification and parameter optimization is saved during the evolutionary parametrization process.

Chapter 3

Technical development

In this chapter we are going to describe the technical development of the software explained with detailed diagrams. First of all we've got to take into account that the project is developed in the Matlab©framework, which implies a procedural paradigm of programming, therefore interaction between functions are described.

3.1 User case diagrams

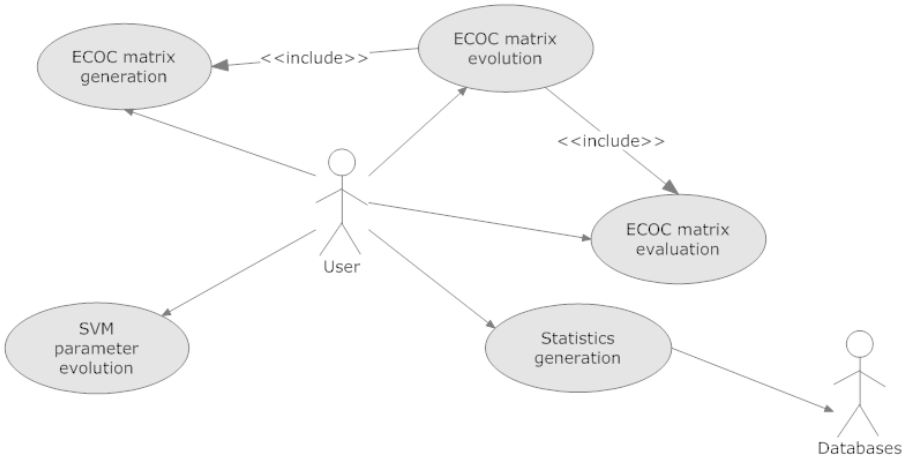


Figure 3.1: User case diagram of the system

3.2 Sequence system diagrams

In next images the DSS diagrams are shown. Each one of the corresponds to a certain system event.

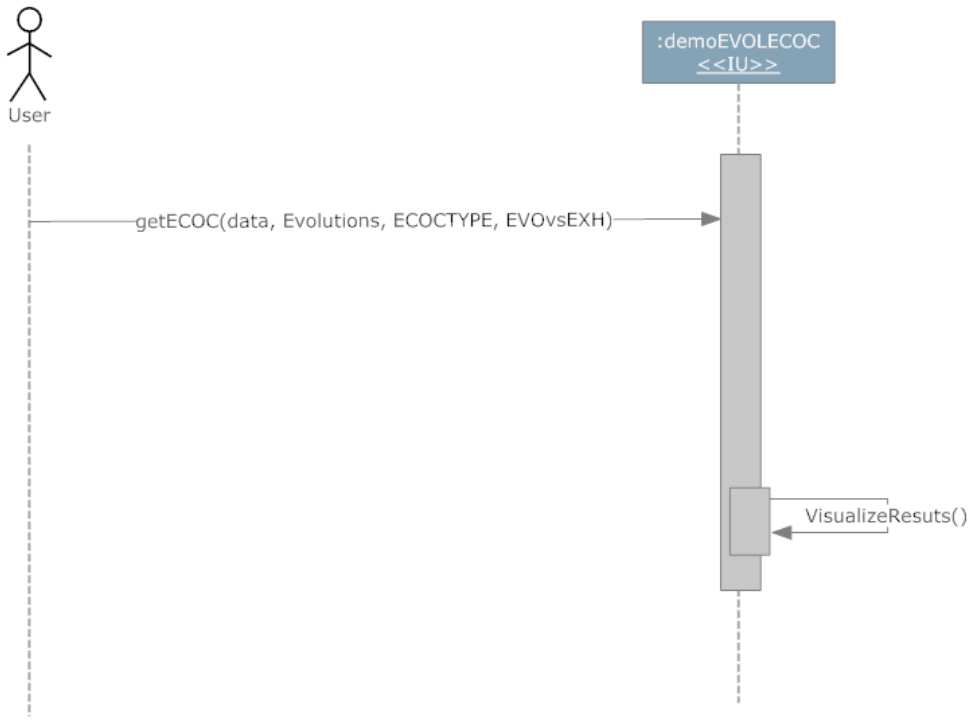


Figure 3.2: DSS_1: User interacting with the user interface

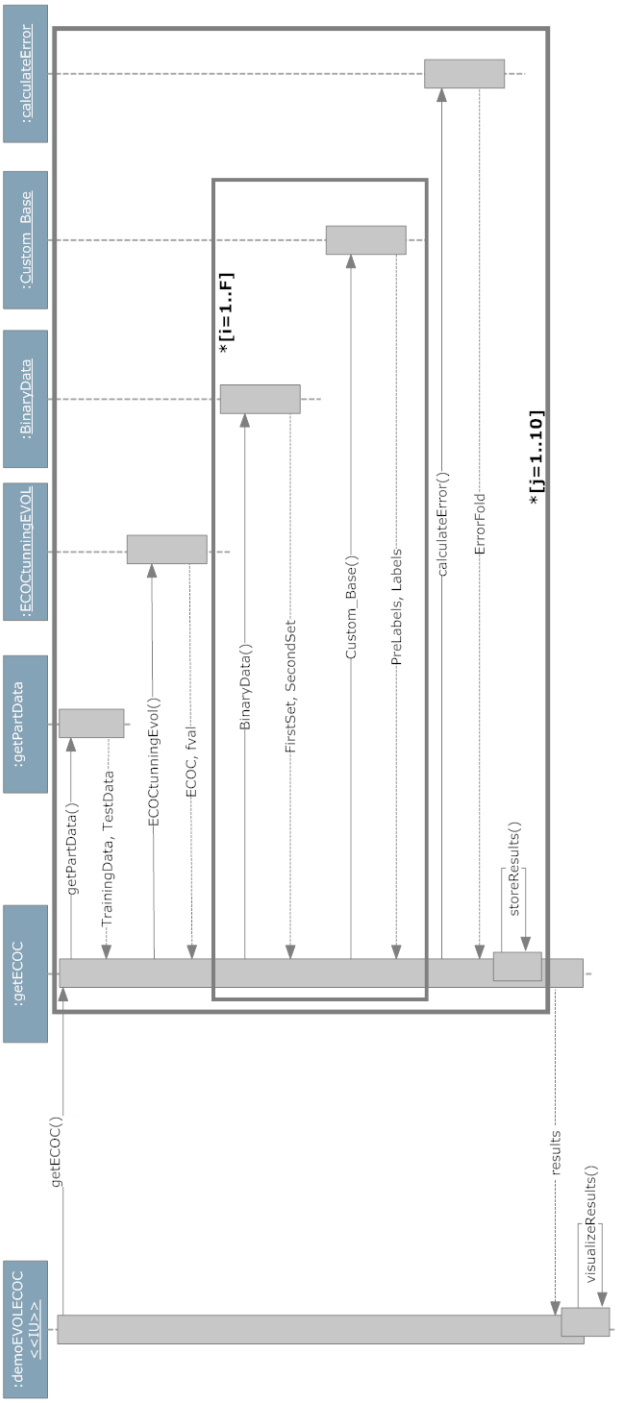


Figure 3.3: DSS_2: Interaction of the interface with the system functions

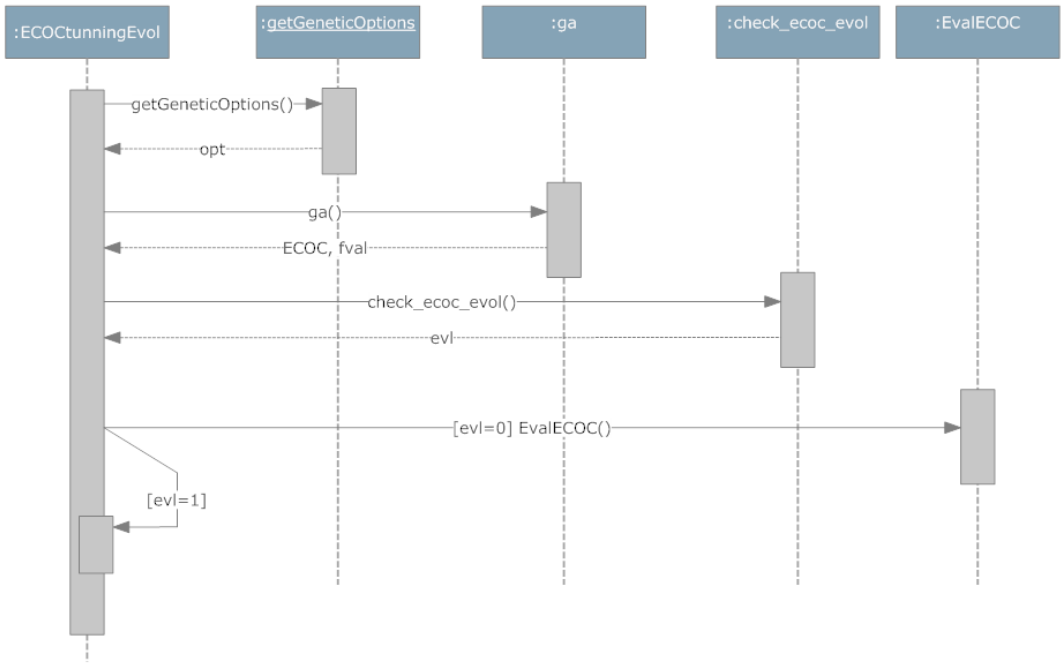


Figure 3.4: DSS_3: Evolution of an ECOC minimal matrix

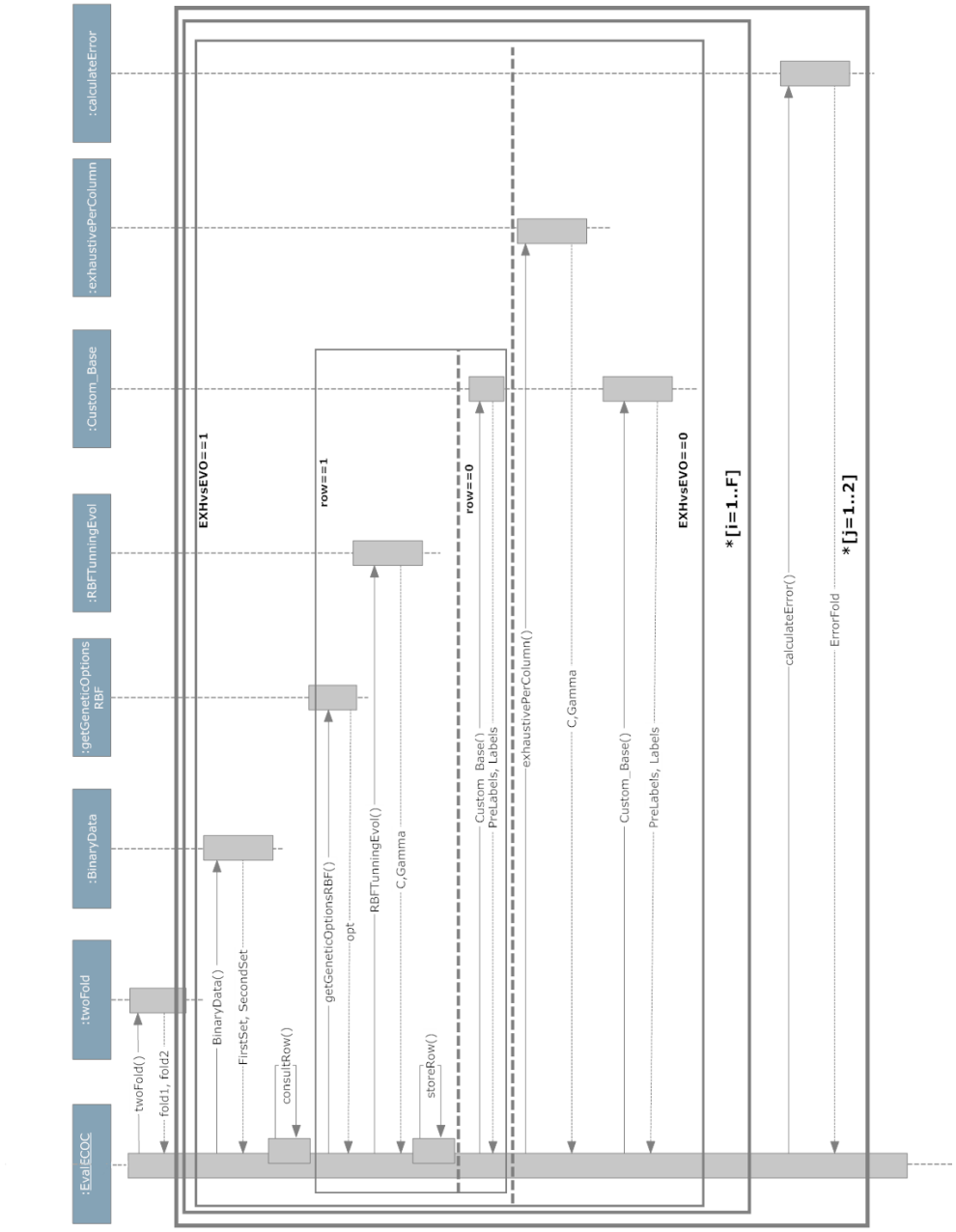


Figure 3.5: DSS_4: Evaluation of an ECOC matrix, with the corresponding evolution of the base classifiers

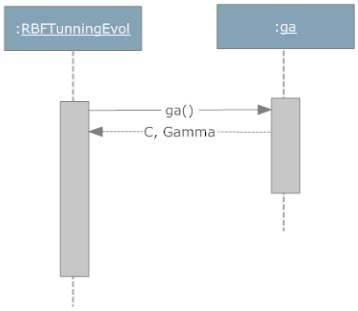


Figure 3.6: DSS_5: Evolutions of the SVM-RBF parameters

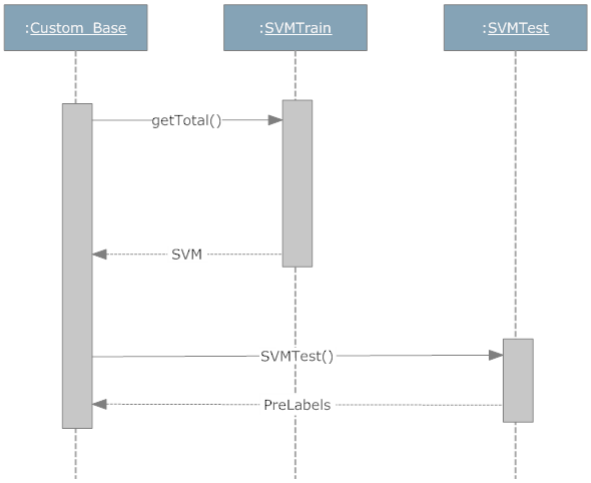


Figure 3.7: DSS_6: Evaluation of a SVM-RBF classifiers with fixed parameters

3.3 Collaboration diagram

In the image is shown the system collaboration diagram. All the function have been implemented in Matlab©, both the ones corresponding to the ECOC designs and the ones corresponding to SVM classifiers (implemented in the OSU-SVM library), for the genetic algorithms the standard Matlab©library is used. The diagram shows the interaction between the functions that implement the system.

As can be seen, different notations are used by different colors. Red functions

correspond to the user interface. Green functions are handlers which yellow functions need. Blue functions are auxiliary functions edited in the same script. Each white function has a concrete functionality and it's isolated in a script

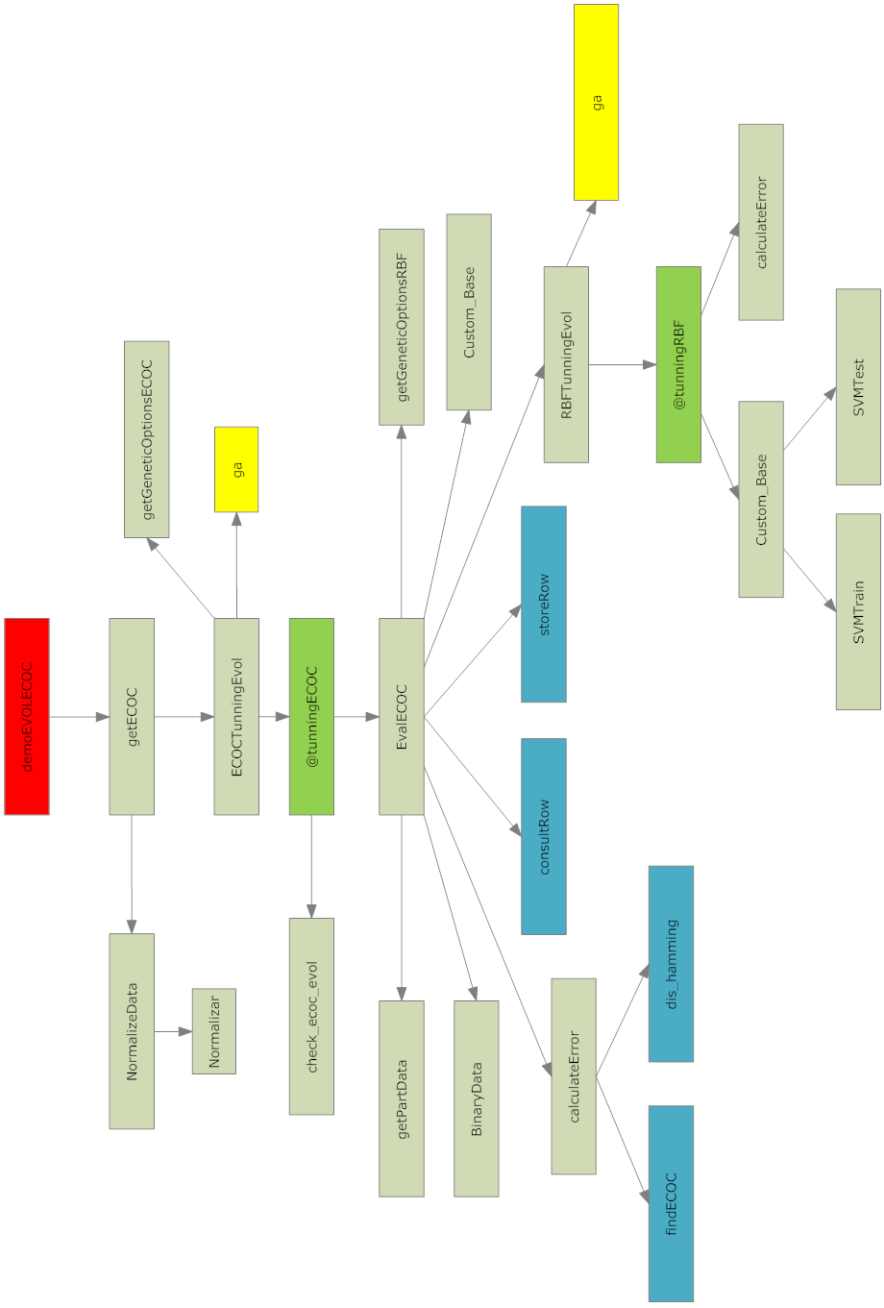


Figure 3.8: Collaboration diagram

3.4 Planning and costs

In this section we are going to describe the planning of the system development as well as the general costs of the creation of the it. In the diagram we can see the time required for every task.

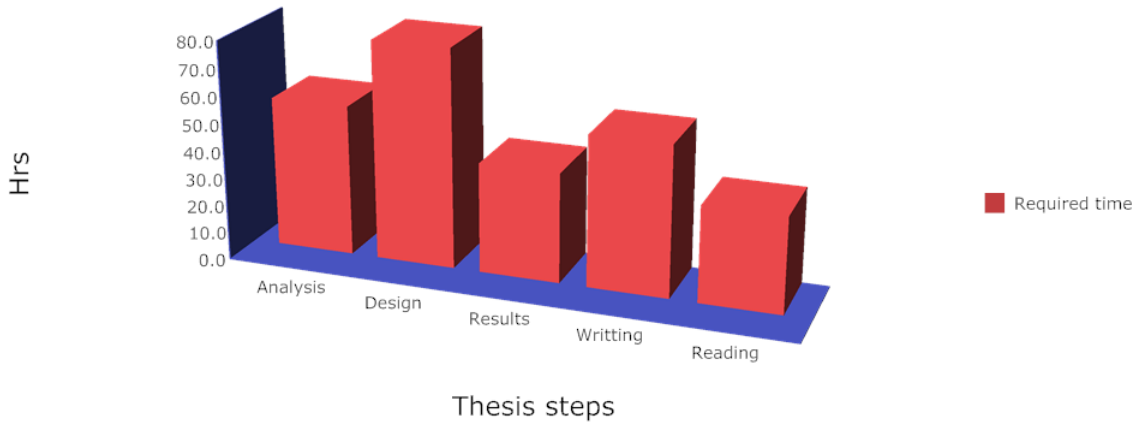


Figure 3.9: Amount of time required by the thesis

In this sense, the plan to follow is the one that can be seen in the image 3.10.

Task	Start	End	Duration	2010					
				January	February	March	April	May	June
Analysis	1/1/2010	2/4/2010	65						
Design	15/2/2010	8/5/2010	60						
Results	1/4/2010	10/6/2010	50						
Witting	1/4/2010	10/6/2010	50						
Reading	20/4/2010	10/6/2010	37						

Figure 3.10: Task planning

In the image can be observed how the most hard tasks where design and writing. This situation is due to the complexity of the study and the academic load that it implied

Now we are going to analyze the costs that would have generated the realization of this project. We have to take into account the license costs and the costs of one analyst programmer per hour. The items that have done, as well as, the necessary hardware. In the table 3.1 costs are shown.

Ítem/Material	Analyst Hours/Material price	Annual license(Euros)	Total (Euros)
Minimal ECOC codign	15	1000	0
OSU-SVM integration	20	0	0
SVM-RBF evolution	20	0	0
ECOC evolution	25	0	0
Test set	25	0	0
Personal computer	800	0	0
External hard drive	200	0	0
TOTAL	100×50 + 800 + 200	1000	7000

Table 3.1: Thesis cost table

Assuming that the price of an analyst programmer is 50 euros per hour, the total cost will be 7000 euros.

Chapter 4

Experimental results

4.1 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 [AN07]. The number of training samples, features, and classes per data set are shown in Table 4.1. Then, we apply the classification methodology in five challenging computer vision categorization problems. First, we use the data of the public *Labeled Faces in the Wild* [HRBM07] dataset to perform the multi-class face classification of a large problem consisting of 184 face categories. Second, we use the video sequences obtained from a Mobile Mapping System [CMP$^+$04] to test the methods in a real traffic sign categorization problem consisting of 36 traffic sign classes. Third, 20 classes from the ARFaces [MB98] data set are classified using the present methodology. Fourth, we classify seven symbols from old scanned music scores, and fifth, we classify the 70 visual object categories from the public MPEG7 data set [mpe].

Table 4.1: UCI repository data sets characteristics.

Problem	#Training samples	#Features	#Classes
Dermathology	366	34	6
Iris	150	4	3
Ecoli	336	8	8
Vehicle	846	18	4
Wine	178	13	3
Segmentation	2310	19	7
Glass	214	9	7
Thyroid	215	5	3
Vowel	990	10	11
Balance	625	4	3
Shuttle	14500	9	7
Yeast	1484	8	10

• *Methods*: We compare the one-versus-one [TR98] and one-versus-all [PGCP65]

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 [DB95]. The ECOC base classifier is the OSU implementation of SVM with Radial Basis Function kernel [SVM03]. 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 [Dem06] in order to look for statistical significance among the obtained performances.

4.1.1 UCI categorization

The classification results obtained for all the UCI data sets considering the different ECOC configurations are shown in Table 4.2. 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 4.2: 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	**97.2±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	**81.3±10.8**	3	79.5±12.2	8	79.2±13.8	28
Vehicle	72.5±14.3	2	82.60±12.4	2	74.2±13.4	4	**83.6±10.5**	6
Wine	95.5±4.3	2	**98.3±2.3**	2	95.5±4.3	3	97.2±2.4	3
Segment	96.6±2.3	3	96.7±1.5	3	96.1±1.8	7	**97.18±1.3**	21
Glass	56.7±23.5	3	51.24±29.7	3	53.85±25.8	6	**60.5±26.9**	15
Thyroid	**96.4±5.3**	2	94.7±5.1	2	95.6±7.4	3	96.1±5.4	3
Vowel	57.7±29.4	3	80.3±11.1	3	**80.7±11.9**	8	78.9±14.2	28
Balance	80.9±11.2	2	87.1±9.2	2	78.9±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	**53.7±11.8**	4	51.1±18.0	10	52.4±20.8	45
Rank & #	2.8	**2.7**	**1.9**	**2.7**	2.9	5.7	2.3	15.9

4.1.2 Computer Vision Applications

In this section, we test the methodology on five challenging Computer Vision problems: faces in the wild, traffic sign, ARface, music scores, and MPEG7 categorization data sets.

Labeled Faces in the Wild categorization

This dataset 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 ten or more examples, having a total of 184 face categories. Finally, in order to extract relevant features from the images, we apply an Incremental Principal Component Analysis procedure [HWZ03], keeping 99.8% of the information. An example of face images is shown in 4.1.

Figure 4.1: Labeled Faces in the Wild dataset.

The results in the first row of Table 4.3 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.

Traffic sign categorization

For this second computer vision experiment, we use the video sequences obtained from the Mobile Mapping System of [CMP+04] to test the ECOC methodology on a real traffic sign categorization problem. In this system, the position and orientation of the different traffic signs are measured with video cameras fixed on a moving vehicle. The system has a stereo pair of calibrated cameras, which are synchronized with a GPS/INS system. The result of the acquisition step is a set of stereo-pairs of images with their position and orientation information. From this system, a set of 36 circular and triangular traffic sign classes are obtained. Some categories from this data set are shown in Figure 4.2. The data set contains a total of 3481 samples of size 32×32, filtered using the Weickert anisotropic filter, masked to exclude the background pixels, and equalized to prevent the effects of illumination changes. These feature vectors are then projected into a 100 feature vector by means of PCA.

The classification results obtained considering the different ECOC configurations are shown in the second row of Table 4.3. The ECOC designs obtain similar classification results with an accuracy upon 90%. However, note that the minimal methodologies use six times less classifiers than the one-versus-all and 105 less times classifiers than the one-versus-one approach, respectively.

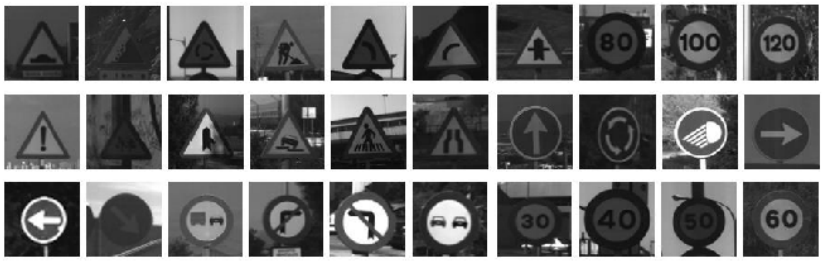


Figure 4.2: Traffic sign classes.

ARFaces classification

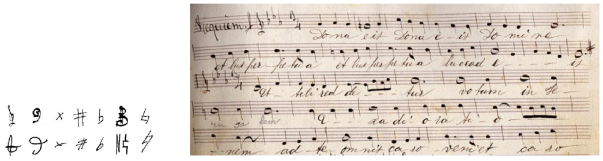
The AR Face database [MB98] is composed of 26 face images from 126 different subjects (70 men and 56 women). The images have uniform white background. The database has from each person two sets of images, acquired in two different sessions, with the following structure: one sample of neutral frontal images, three samples with strong changes in the illumination, two samples with occlusions (scarf and glasses), four images combining occlusions and illumination changes, and three samples with gesture effects. One example of each type is plotted in Figure 4.3. For this experiment, we selected all the samples from 20 different categories (persons).

Figure 4.3: ARFaces data set classes. Examples from a category with neutral, smile, anger, scream expressions, wearing sun glasses, wearing sunglasses and left light on, wearing sun glasses and right light on, wearing scarf, wearing scarf and left light on, and wearing scarf and right light on.

The classification results obtained considering the different ECOC configurations are shown in the third row of Table 4.3. In this case, the one-versus-one strategy obtains significant superior results to the rest of approaches, and the Evolutionary Minimal approaches clearly outperforms the one-versus-all ECOC results.

Clefs and accidental data set categorization

The data set of clefs and accidental is obtained from a collection of modern and old musical scores (19th century) of the Archive of the Seminar of Barcelona. The data set contains a total of 4098 samples among seven different types of clefs and accidental from 24 different authors. The images have been obtained from original image documents using a semi-supervised segmentation approach [FLS06]. The main difficulty of this data set is the lack 389 of a clear class separability because of the variation of writer styles and the absence of a standard notation. A pair of segmented samples for each of the seven classes showing the high variability of clefs and accidental appearance from different authors can be observed in Figure 4.4(a). An example of an old musical score used to obtain the data samples are shown in Figure 4.4(b). The object images are described using the Blurred Shape Model descriptor (BSM).



(a) (b)

Figure 4.4: (a) Object samples, (b) Old music score.

The classification results obtained considering the different ECOC configurations are shown in the fourth row of Table 4.3. In this case, the results are also comparable for all the strategies, with accuracies upon 80%.

MPEG7 categorization

The MPEG7 data set contains 70 classes with 20 instances per class, which represents a total of 1400 object images. All samples are described using the Blurred Shape Model descriptor. A couple of samples for some categories of this data set are shown in Figure 4.5.

The classification results obtained considering the different ECOC configurations are shown in the fifth row of Table 4.3. These results are very satisfactory since one can see very similar results between the evolutionaries and one-versus-one strategies, taking into account that the last approach requires near 350 times the number of classifiers required by our proposal.

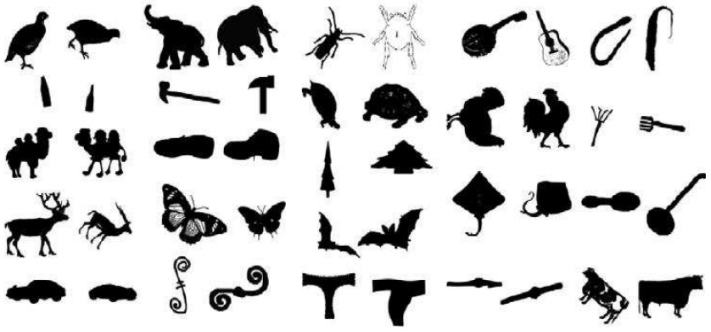


Figure 4.5: MPEG7 samples.

Table 4.3: Computer Vision data sets 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	8	**31.7±2.3**	8	30.02.4±	8	25.0±3.1	184	-	16836
Traffic	90.8±4.1	6	90.6±3.4	6	90.7±3.7	6	**91.8±4.6**	36	90.6±4.1	630
ARFaces	76.0±7.2	5	85.84.0±5.2	5	84.2±5.3	5	84.0±6.3	20	**96.0±2.5**	190
Clefs	81.2±4.2	3	**95.6±9.3**	3	81.7±8.2	3	80.8±11.2	7	84.2±6.8	21
MPEG7	89.29±5.1	7	90.4±4.5	7	90.1±4.9	7	87.8±6.4	70	**92.8±3.7**	2415
Rank & #	3.6	**6.2**	2.0	**6.2**	2.8	**6.2**	3.8	148.6	**1.75**	37800

Chapter 5

Evaluation of results and discussion

5.1 UCI Machine Learning repository results

In order to analyze if the difference between method ranks is statistically significant, we apply a statistical test. In order to reject the null hypothesis that the measured ranks differ from the mean rank, and that the ranks are affected by randomness in the results, we use the Friedman test. The Friedman statistic value is computed as follows:

$$X_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{5.1}$$

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:

$$F_F = \frac{(N-1)X_F^2}{N(k-1) - X_F^2} \tag{5.2}$$

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, increasing the generalization capability of the system. 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 available for optimization.

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 4.2. 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 five challenging Computer Vision problems with several object categories.

5.2 Computer Vision results

Analyzing globally the results of the Computer Vision classification problems, which mean ranks are shown in the last row of Table 4.3, one can see that globally, the one-versus-one is the first choice, followed by the evolutionary proposals. The last two positions are for the binary and one-versus-all coding designs.

In this case, applying the Friedman statistic, we obtain a value of $X_F^2 = -3.71$, and a corrected value of $F_F = -0.62$. With five methods and five Computer Vision experiments, F_F is distributed according to the F distribution with 4 and 16 degrees of freedom. The critical value of $F(4, 16)$ for 0.05 is 3.01. As the value of F_F is no higher than 3.01 we can state that there are no statistical different among the ECOC performances. This means that all five strategies are suitable in order to deal with multi-class Computer Vision problems with several categories. This result also encourages the use of the Minimal approach. Note that for example, in the Faces in the Wild experiment the Minimal ECOC approach requires 10 classifiers in comparison with the 185745 classifiers required by the one-versus-one ECOC strategy.

5.3 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 five multi-class Computer Vision problems with several object categories show that the proposed methodology obtains statistically equivalent results than 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.

Bibliography

[ACSS02] N.E. Ayat, M. Cheriet, C.Y. Suen, and M. Cheriet C. Y. Suen. Empirical error based optimization of svm kernels: Application to digit image recognition. In *In the 8 th IWFHR, Niagaraon-the-lake*, pages 105–110. Springer Verlag, 2002.

[AN07] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[ASS02] E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *JMLR*, volume 1, pages 113–141, 2002.

[BC95] Shumeet Baluja and Rich Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russel, editors, *The Int. Conf. on Machine Learning 1995*, pages 38–46, San Mateo, CA, 1995. Morgan Kaufmann Publishers.

[BZ09] T Aráz E. Buck and Bin Zhang. Svm kernel optimization: An example in yeast protein subcellular localization prediction. 2009.

[CMP+04] J. Casacuberta, J. Miranda, M. Pla, S. Sanchez, A.Serra, and J.Talaya. On the accuracy and performance of the GeoMobil system. In *International Society for Photogrammetry and Remote Sensing*, 2004.

[CS02] K. Crammer and Y. Singer. On the learnability and design of output codes for multi-class problems. In *Machine Learning*, volume 47, pages 201–233, 2002.

[Dar] Charles Darwin. *The Origin of Species*. Gramercy.

[DB95] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. In *JAIR*, volume 2, pages 263–286, 1995.

[Dem06] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7:1–30, 2006.

[DK95] T. Dietterich and E. Kong. Error-correcting output codes corrects bias and variance. In ICML, editor, *S. Prieditis and S. Russell*, pages 313–321, 1995.

[EPP09] S. Escalera, O. Pujol, and P.Radeva. On the decoding process in ternary error-correcting output codes. *Transactions in Pattern Analysis and Machine Intelligence*, 99(1), 2009.

[EPP10] S. Escalera, O. Pujol, and P.Radeva. Error-correcting output codes library. *Journal of Machine Learning Research*, 11:661–664, 2010.

[FHT98] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Dept. of Statistics, Stanford University Technical Report., 1998.

[FLS06] A. Fornés, J. Lladós, and G. Sánchez. Primitive segmentation in old handwritten music scores. *Graphics Recognition*, 3926:279–290, 2006.

[Gha02] Rayid Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *In Proceedings of the International Conference on Machine Learning*, pages 187–194, 2002.

[Hol75] J.H. Holland. *Adaptation in natural and artificial systems: An analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[HRBM07] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik L. Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report University of Massachusets Amherst, 07-49, October 2007.

[HWZ03] W Hwang, J. Weng, and Y. Zhang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.

[KGWM01] J Kittler, R Ghaderi, T Windeatt, and J Matas. Face verification using error correcting output codes. In *In Computer Vision and Pattern Recognition CVPR01*, pages 755–760. IEEE Press, 2001.

[LdC08] Ana Carolina Lorena and André C.P.L.F. de Carvalho. Evolutionary tuning of svm parameter values in multiclass problems. *Neurocomputing*, 71(16-18):3326 – 3334, 2008.

[MB98] A. Martinez and R. Benavente. The AR face database. In *Computer Vision Center Technical Report #24*, 1998.

[mpe] http://www.cis.temple.edu/latecki/research.html.

[Nic03] Tyrone Nicholas. Using adaboost and decision stumps to identify spam e-mail, 2003.

[PGCP65] Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz. Learning machines. In *McGraw-Hill*, 1965.

[Pol06] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[PPF04] A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. In *Neural Networks*, volume 15, pages 45–54, 2004.

[PRV06] Oriol Pujol, Petia Radeva, and Jordi Vitria. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1007–1012, 2006.

[RK04a] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, 2004.

[RK04b] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *JMLR*, 5:101–141, 2004.

[SVM03] OSU SVM. Osu svm classifier matlab toolbox, 2003.

[TR98] T.Hastie and R.Tibshirani. Classification by pairwise grouping. *NIPS*, 26:451–471, 1998.

[UW04] W. Utschick and W. Weichselberger. Stochastic organization of output codes in multiclass learning problems. In *Neural Computation*, volume 13, pages 1065–1102, 2004.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. pages 511–518, 2001.