This document is the Accepted Manuscript version of a Published Work that appeared in final form in **Neurocomputing** 171, 1576-1590 (2016), copyright © 2016 Elsevier. To access the final edited and published work see `https://doi.org/10.1016/j.neucom.2015.07.078`

Undirected Cyclic Graph Based Multiclass Pair-wise Classifier: classifier number reduction maintaining accuracy

I. Mendialdua[a,*], G. Echegaray[b], I. Rodriguez[a], E. Lazkano[a], B. Sierra[a]

[a]*Department of Computer Science and artificial Intelligence*
University of the Basque Country UPV/EHU
Donostia-San Sebastian 20018, Spain.
[b]*Applied Mathematics Department*
University of the Basque Country UPV/EHU
Donostia-San Sebastian 20018, Spain.

Abstract

Supervised Classification approaches try to classify correctly the new unlabelled examples based on a set of well-labelled samples. Nevertheless, some classification methods were formulated for binary classification problems and has difficulties for multi-class problems. Binarization strategies decompose the original multi-class dataset into multiple two-class subsets. For each new sub-problem a classifier is constructed. One-vs-One is a popular decomposition strategy that in each sub-problem discriminates the cases that belong to a pair of classes, ignoring the remaining ones. One of its drawbacks is that it creates a large number of classifiers, and some of them are irrelevant. In order to reduce the number of classifiers, in this paper we propose a new method called Decision Undirected Cyclic Graph. Instead of making the comparisons of all the pair of classes, each class is compared only with other two classes; evolutionary computation is used in the proposed approach in order to obtain suitable class pairing. In order to empirically show the performance of the proposed approach, a set of experiments over four popular Machine Learning algorithms are carried out, where our new method is compared with other well-known decomposition strategies of the literature obtaining promising results.

Keywords: Machine Learning, Supervised Classification, Decomposition Strategies, One-vs-One

1. Introduction

In supervised classification the goal is to build a classifier which given a new case, makes a prediction about the class to which the new observation belongs. To do so, the supervised classification paradigms requires a training set, i.e. a collection of well classified samples. Let $TR = \{x_i, \theta_i\}_{i=1}^N$ be the training set of N well labeled examples, where x_i represents i-th individual feature vector, and θ_i is the class the individual belongs to. Based on the training set the supervised classification builds a general rule, also called as classifier, that is used to predict the class of the new unlabelled case.

Although many real world problems are multi-class problems, some kind of approaches, such as SVM, has difficulties to build a classifier to distinguish between more than two classes. In order to solve this problem Class Binarization strategies were proposed. Class Binarization strategies decompose the original multi-class problem into many binary classification sub-problems. In each sub-problem the classes are decoded with 3 possible values {-1,0,+1} and a classifier is constructed to differentiate between positive and negative values; normally the same base classifier is used in all the sub-problems. These techniques are two-step methods: in the first step a classifier is learned for each binary sub-problem, and in the second step the outputs of these binary classifiers are combined to obtain the final prediction.

[*]Corresponding author at: Robotics and Autonomous System Research Group, University of the Basque Country UPV/EHU, Donostia-San Sebastian 20018, Spain

Email address: `inigo.mendialdua@ehu.es` (I. Mendialdua)

In the specialized literature three main Class Binarization strategies can be found: ”One vs One” (OVO), ”One vs All” (OVA) and “Error Correcting Output Codes” (ECOC).

- One vs All (OVA) [1]: In each sub-problem one class is compared with the rest of classes.
- One vs One (OVO) [12]: In each sub-problem only the cases belonging to two classes are compared between them, ignoring the remaining ones.
- Error Correcting Output Codes (ECOC) [8]: In each sub-problem all the classes are grouped into two groups, and the two groups are compared between them.

Among these three strategies OVO is which more attention has received in the literature. Some proposals try to improve the combination of the outputs, while other approaches try to solve some of the disadvantages of OVO. One of its main drawbacks is the number of sub-problems that OVO needs. Many of the binary classifiers are irrelevant and are forced to give wrong answers for many instances, because each binary classifier must classify every pattern with one of the two classes used in its training set. If a pattern belongs to class i, all the classifiers that are not trained to differentiate this class will cast wrong votes

In this paper our aim is to present a novel strategy which reduces the number of classifiers in OVO in the classification phase. Instead of being compared with all the other classes, each class is only compared with other 2 classes. We represent our method as an undirected cyclic graph or a list, that is why we call it Decision Undirected Cyclic Graph (DUCG). In order to find the best ordering of the list we have used an evolutionary computation approach obtained from the state-of-the-art called Edge Histogram-Based Sampling Algorithm (EHBSA) [35]. To show the behaviour of our proposal, we have compared it with other Class Binarization strategies over 27 UCI databases. We have carried out these experiments over 4 well known Machine Learning methods: SVM, C4.5 Decision Tree, Ripper and Multilayer Perceptron. Two performance measures have been used to evaluate the results: Classification rate and Cohen’s Kappa. The obtained results show competitive performance of our proposal, specially in the problems with a large number of classes.

The rest of the paper is organized as follows. In Section 2 we review the decomposition techniques, with special attention to OVO and OVA strategies. Section 3 describes the proposed approach and Section 4 shows the experimental results obtained. Finally, Section 5 states the conclusions of our work and future research lines.

2. Class Binarziation

Class Binarization is performed in two steps: decomposition and combination.

The decomposition step consists of dividing the K class problem into several binary sub-problems. The most popular strategy is to divide the classes into two groups; in this way the binary classifier distinguishes the classes of one group with the classes of the other group. Commonly the code-matrix is used to represent how the classes are grouped.

Figure 1 illustrates a code-matrix example: each row represents a class and each column represents a binary classifier. Each class takes values in the set {-1,0,+1}, where +1 indicates the classes associated to the positive-class, -1 indicates the classes associated to the negative-class and 0 indicates that the class is ignored for this binary problem. Figure 1 illustrates an example of a decomposition of a 5-class problem $\{\theta_1,\theta_2,\theta_3,\theta_4,\theta_5\}$ into 6 binary sub-problems $\{f_1,f_2,f_3,f_4,f_5,f_6\}$. For instance, it can be seen that the classifier f_1 is constructed in such a manner that the cases belonging to θ_1 and θ_2 are grouped in class +1 and the cases belonging to θ_3 and θ_5 in class -1. So this classifier compares θ_1 and θ_2 classes with θ_3 and θ_5, while the cases that belong to θ_4 are ignored.

In classification time, each binary classifier returns a prediction. So the combination step consists of combining these predictions. Therefore, once the decomposition strategy is fixed, it is crucial to select a proper combination of the outputs in order to make the final prediction.

Different decomposition strategies have been developed. Two of the most popular are OVA and OVO, which are described next.

$$
\text{classes}\ \left\{\begin{array}{c} \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{array}\right.
\begin{array}{c} \overbrace{\begin{array}{cccccc} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \end{array}}^{classifiers} \\
\begin{pmatrix} +1 & 0 & -1 & -1 & 0 & +1 \\ +1 & +1 & -1 & -1 & +1 & 0 \\ -1 & +1 & +1 & -1 & 0 & 0 \\ 0 & -1 & 0 & +1 & 0 & +1 \\ -1 & -1 & 0 & -1 & -1 & -1 \end{pmatrix} \end{array}
\qquad
\begin{array}{l} f_1 \rightarrow \theta_1, \theta_2\ vs\ \theta_3, \theta_5 \\ f_2 \rightarrow \theta_2, \theta_3\ vs\ \theta_4, \theta_5 \\ f_3 \rightarrow \theta_3\ vs\ \theta_1, \theta_2 \\ f_4 \rightarrow \theta_4\ vs\ \theta_1, \theta_2, \theta_3, \theta_5 \\ f_5 \rightarrow \theta_2\ vs\ \theta_5 \\ f_6 \rightarrow \theta_1, \theta_4\ vs\ \theta_5 \end{array}
$$

Figure 1: Example of a code matrix

2.1. One Vs All (OVA)

OVA decomposition scheme divides a K class multi-class problem into K two-class problems, where in each binary sub-problem a single class is separated from all other classes.

In Figure 2(a) OVA's code matrix for 4 classes can be seen: in each classifier one class is represented as positive class while all the other 3 classes are represented as negative-class.

$$
\begin{array}{cc}
\begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{pmatrix} &
\begin{pmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix} \\
\text{(a) One Vs All} & \text{(b) One Vs One}
\end{array}
$$

Figure 2: OVA and OVO code-matrix

One of the disadvantages of OVA is that most of the binary sub-problems are unbalanced. As one class is compared with all the other classes, it is common that all sub-problems return a class-negative prediction, hence is obtained a tie between all the classes when the majority vote is used. Due to that problem, it is common to select the class with the highest confidence level as a final prediction.

2.2. One Vs One (OVO)

In OVO the original K class multi-class problem, $\theta_1, ..., \theta_K$, is divided into $K(K-1)/2$ two-class sub-problems. In each sub-problem a classifier is learnt using only the cases that belong to a pair of classes (θ_i,θ_j), where $\theta_i \neq \theta_j$; the remaining cases are ignored.

Figure 2(b) illustrates a code-matrix of OVO for 4 classes: in each classifier one class is represented as +1 class, another one is represented as -1 and the remaining two classes are represented as 0.

Different aggregations of OVO are proposed in the literature to combine the outputs of the sub-problems. The simplest combination strategy is the majority vote, where each output gives a vote for a class and that class which obtains the largest number of votes is returned [12] [11]. An immediate extension is the Weighted Voting strategy: to use the confidence level of each base classifier as a vote [23]. Hastie and Tibshirani [20] present another combination where they try to find the best approximation of the class posterior probabilities given the posterior probabilities of the pairwise sub-problems. Wu et al. [36] also estimate the posterior probabilities of each class, but the optimization formulation is different from [20].

One of the disadvantages of OVO is the number of sub-problems that it creates. It is worth mentioning that most of them are irrelevant and they return wrong answers for many instances: if an instance belongs to class θ_i, all the classifiers that are not trained to differentiate θ_i will return wrong predictions. On the other hand, one of the advantages of OVO is that these sub-problems are constructed with fewer examples and thus has more freedom for fitting a decision boundary between two classes.

2.3. Related Works

Various popular machine learning techniques, such as Support Vector Machines (SVM), were originally conceived for the solution of two-class classification problems. As a consequence, they were not able to solve multi-class problems. Therefore, in order to deal with this problem the first Class Binarization problems were proposed, and due to the promising results obtained, this strategies has been extended to other kind of classifiers, like Ripper [12] and C4.5 [8].

In several works different Class Binarization strategies have been compared. Some of them conclude that OVO is significantly better than OVA[12] [22]. However, Rifkin and Klautau [33] suggest that when the binary classifiers are well-tuned, OVA performs as well as the other strategies. Recently two empirical studies have appeared concerning to this question [13] [17]. Galar et al. [13] compare different OVO and OVA strategies. While García-Pedrajas and Ortiz-Boyer [17] compare the different Class Binarization strategies among them. They consider that OVO is the best choice when weak classifiers are used, while ECOC is recommended with powerful learners. Moreover, they show that when ECOC uses the same number of classifiers as OVO ($K(K-1)/2$), OVO obtains a slight advantage.

Among the Class Binarization strategies, OVA is which less attention has received in the literature, and there are not many aggregations. Hong et al. [21] propose integrate Naive Bayes in OVA to order dynamically the sequence of the classifiers. On the other hand, Kumar and Gopal [26] propose a method where they reduce the number of samples of the classifier discarding the instances that are out of a established region.

2.4. Reducing the number of classifiers

Some works try to reduce the number of sub-problems in OVO. Among those works one of the most popular method is the Decision Directed Acyclic Graph (DDAG) [29]. This method constructs a rooted binary acyclic graph. In each level a classifier discriminates between two classes and the selected class is compared with another class in the next level. In this way they reduce the number of sub-problems to $K-1$. One of the disadvantages of this method is that the classes compared in the first level are less likely to be predicted than the classes compared in the last levels. Various versions of this method have been proposed in the literature; one of the most famous is the so called ADAG [24].

Other techniques also are based on a hierarchical structure: Fei and Liu [9] introduce a binary tree where in each node two or more classes are distinguished, Lorena and Carvalho [28] propose to use 4 different separability criteria and they use Kruskal algorithm to generate a tree of binary classifiers, Pujol et. al. [31] use Mutual Information for class separation, Ghaffari and Yazdi [18] use divisive clustering for class partitioning and Kumar et. al. [27] also use clustering, at the same time in each node a feature extractor is applied in order to maximize the discrimination between meta-classes.

García-Pedrajas and Ortiz-Boyer [16] and Ko [25] present independently a combination of OVA and OVO. Firstly they apply OVA. Next they select the two classes with the highest confidence level. And finally OVO is applied with these two classes. Then only $K+1$ classifiers have to be used in the classification process. This method is called All-And-One (A&O).

On the other hand Galar et. al. [14] and Bagheri et. al [2] present a similar idea: they suggest to use the dynamic classifier selection for OVO in order to avoid the non-competent classifiers. The K nearest neighbors of the instance to be classified are obtained and the classes that appear in this neighborhood are considered as the probable classes. With these most probable classes OVO is applied ignoring the remaining ones.

Bautista et. al. [3] propose to create the minimal ECOC. They try to find the minimal ECOC using Evolutionary Computation, at the same time they try to find the best parameters of each classifier.

3. Proposed Approach: Decision Undirected Cyclic Graph (DUCG)

As mentioned in previous sections one of the disadvantages of OVO is the large amount of classifiers that it builds. In order to avoid it, DDAG method was proposed, but this algorithm implies another problem: the classes that are compared first are less likely to be predicted because they have to be selected in all the comparisons.

In order to avoid these weaknesses we propose a new method called Decision Undirected Cyclic Graph (DUCG), where the classes are compared in pairs, as in OVO, but instead of performing all the comparisons, each class is compared only with other two classes. This way permits to reduce the amount of binary classifiers and the same chance is given to all the classes. Although the use of all pair comparisons seems to be more effective, our believe is that selecting the proper comparisons the accuracy can be improved.

DUCG can be represented as a cycle graph: a single graph where the number of nodes and edges are the same and every node has degree 2. Figure 3 shows an illustrative example of 6 classes where each node corresponds to a class and the edges denote the pairwise comparisons of the classes. It can be seen that our method compares only 6 pair of classes, ignoring the remaining comparisons.

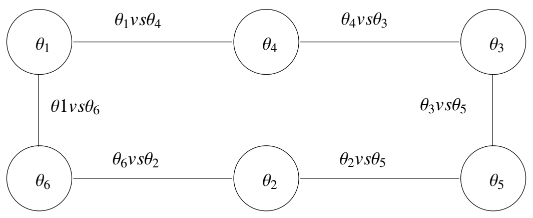
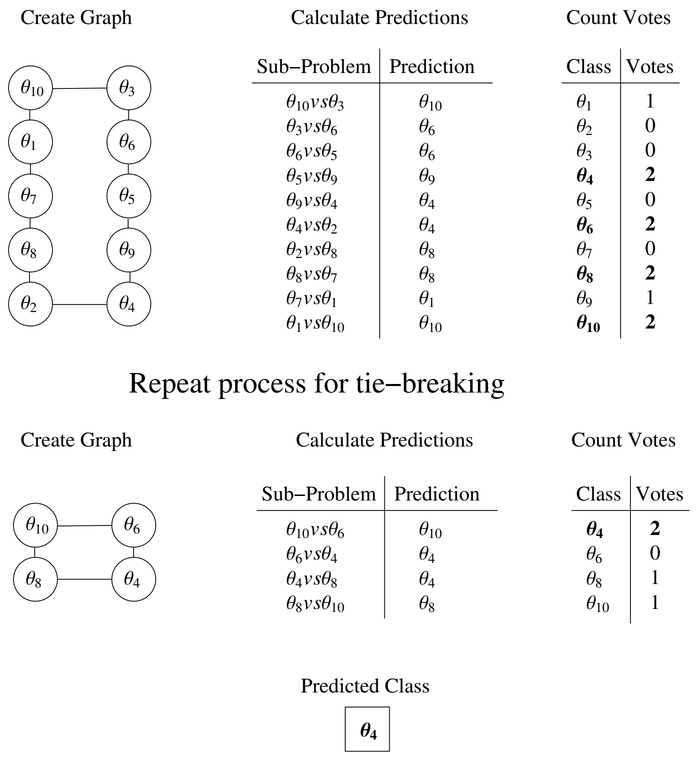


Figure 3: Example of the structure of DUCG for a 6-class problem

It is worth mentioning that as in our method is common to be ties (each class obtains at most 2 votes), DUCG is applied recursively considering only the tie-classes.

In order to give a better explanation of how DUCG works, in Figure 4 an example of a 10-class problem is illustrated. Firstly our method creates the graph to decide the pairwise comparisons. Each sub-problem returns a prediction and then the number of votes that each class receives are computed. It can be seen in the example that there are 4 classes that receive 2 votes (the maximum they can receive). In order to break the ties, our method repeats the process only considering those 4 classes. The new graph is created, the sub-problems return the predictions and the votes are counted. This time there is only one class that receives 2 votes, thus DUCG assigns this class to the new instance.



Sub−Problem	Prediction
$\theta_{10} vs \theta_3$	θ_{10}
$\theta_3 vs \theta_6$	θ_6
$\theta_6 vs \theta_5$	θ_6
$\theta_5 vs \theta_9$	θ_9
$\theta_9 vs \theta_4$	θ_4
$\theta_4 vs \theta_2$	θ_4
$\theta_2 vs \theta_8$	θ_8
$\theta_8 vs \theta_7$	θ_8
$\theta_7 vs \theta_1$	θ_1
$\theta_1 vs \theta_{10}$	θ_{10}

Class	Votes
θ_1	1
θ_2	0
θ_3	0
$\boldsymbol{\theta_4}$	**2**
θ_5	0
$\boldsymbol{\theta_6}$	**2**
θ_7	0
$\boldsymbol{\theta_8}$	**2**
θ_9	1
$\boldsymbol{\theta_{10}}$	**2**

Sub−Problem	Prediction
$\theta_{10} vs \theta_6$	θ_{10}
$\theta_6 vs \theta_4$	θ_4
$\theta_4 vs \theta_8$	θ_4
$\theta_8 vs \theta_{10}$	θ_8

Class	Votes
$\boldsymbol{\theta_4}$	**2**
θ_6	0
θ_8	1
θ_{10}	1

Figure 4: Illustrative example of DUCG for a 10-class problem

$$\begin{aligned} S_1 &= (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5) \\ S_2 &= (\theta_4, \theta_2, \theta_5, \theta_1, \theta_3) \\ S_3 &= (\theta_4, \theta_5, \theta_1, \theta_3, \theta_2) \\ S_4 &= (\theta_3, \theta_4, \theta_2, \theta_1, \theta_5) \\ S_5 &= (\theta_4, \theta_2, \theta_1, \theta_3, \theta_5) \\ S_6 &= (\theta_5, \theta_2, \theta_3, \theta_4, \theta_1) \end{aligned}$$

(a) Permutations

$$\begin{pmatrix} 0 & 3.1 & 3.1 & 1.1 & 5.1 \\ 3.1 & 0 & 3.1 & 4.1 & 2.1 \\ 3.1 & 3.1 & 0 & 4.1 & 2.1 \\ 1.1 & 4.1 & 4.1 & 0 & 3.1 \\ 5.1 & 2.1 & 2.1 & 3.1 & 0 \end{pmatrix}$$

(b) EHM Adjacency

$$\begin{pmatrix} - & 0.25 & 0.25 & 0.09 & 0.41 \\ 0.25 & - & 0.25 & 0.33 & 0.17 \\ 0.25 & 0.25 & - & 0.33 & 0.17 \\ 0.09 & 0.33 & 0.33 & - & 0.25 \\ 0.41 & 0.17 & 0.17 & 0.25 & - \end{pmatrix}$$

(c) EHM Normalized

Figure 5: Example of Edge Histogram Matrix

3.1. Build the graph

The pairwise organization of the classifiers can also be seen as a list where each class is compared with the classes that are next to it. Moreover, comparing the last class with the first in the list a cyclic solution is obtained, for instance $(\theta_1, \theta_4, \theta_3, \theta_5, \theta_2, \theta_6)$ is equivalent to the graph of Figure 3. Since our aim is to find the best ordering of classes we treat our problem as a permutation-based problem.

There exist many combinational problems whose solutions can be naturally represented as permutations. However, the meaning of these permutations can vary throughout the problems. In our particular case, our problem can be considered similar to the Travelling Salesman Problem (TSP). TSP is a problem where the solutions are cyclic and the relevant information is given by the relative ordering of the classes in the permutation. The information drawn from the absolute positions of each class is not meaningful. For instance, $\sigma = (\theta_1, \theta_3, \theta_2, \theta_4)$ and $\sigma' = (\theta_2, \theta_3, \theta_1, \theta_4)$ represent the same solution since both make the same classes comparisons: $[\theta_1 vs \theta_3, \theta_1 vs \theta_4, \theta_2 vs \theta_3, \theta_2 vs \theta_4]$. Thus, the search space of the solutions is reduced from K! to K!/2K.

As we mentioned before, the base classifier can have difficulties to differentiate some pairs of classes, so our aim is to avoid them. So in a validation phase we try to find the best combination of two-class comparisons. If the number of classes is low, the treatment of all the candidate-solutions is possible, but while the number of classes increases the computational cost is higher and it could become unaffordable. Because of that fact, this problem can be considered as an optimization design process. One promising strategy for this optimization issue is to use an evolutionary algorithm-based approach. Recently, some of the most well-known evolutionary algorithms used for the permutation problems are based on the Estimation of Distribution Algorithms (EDA). EDAs combine statistical learning with population-based search in order to automatically identify and exploit certain structural properties of optimization problems.

Recently, Ceberio et al. [5] have carried out a review of state-of-the-art EDAs applied to permutation-based problems and they concluded that Edge Histogram-Based Sampling Algorithm (EHBSA) [35] is the most successful proposal to solve the TSP.

3.1.1. Edge Histogram-Based Sampling Algorithm (EHBSA)

Given a sample of solutions, EHBSA estimates a bi-variate probabilistic model which learns the pairwise adjacency of the items within the permutation.

The algorithm starts by generating a random population of samples and the best solutions are selected. In the next step, an Edge Histogram Matrix (EHM) for the selected solutions is constructed. Based on EHM, new solutions are generated. Some of the old solutions are replaced by the new ones and the process is repeated until the termination criteria is met.

EHM counts the number of times that two items are next to each other in the given sample of solutions. In Figure 5 an example of the construction of an EHM given 6 permutations of 5 classes $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ is illustrated. In order to avoid probability 0, in Figure 5(b) an ϵ value is added to the sum of the adjacency. Normalizing the rows of Figure 5(b) the probabilities of adjacency are obtained, which are shown in Figure 5(c).

Based on EHM, EHBSA generates new solutions following the next procedure:

1. The class of the first position is fixed randomly.
2. To sample the next positions
   (a) Discard previously sampled classes of EHM

$$S' = (\theta_4,)$$

$$\begin{pmatrix} - & 0.25 & 0.25 & 0.09 & 0.41 \\ 0.25 & - & 0.25 & 0.33 & 0.17 \\ 0.25 & 0.25 & - & 0.33 & 0.17 \\ \multicolumn{5}{c}{\boxed{0.09 \quad 0.33 \quad 0.33 \quad - \quad 0.25}} \\ 0.41 & 0.17 & 0.17 & 0.25 & - \end{pmatrix}$$

(a) Step 1

$$S' = (\theta_4, \theta_3)$$

$$\begin{pmatrix} - & 0.27 & 0.27 & - & 0.45 \\ 0.37 & - & 0.37 & - & 0.25 \\ \multicolumn{5}{c}{\boxed{0.37 \quad 0.37 \quad - \quad - \quad 0.25}} \\ - & - & - & - & - \\ 0.55 & 0.22 & 0.22 & - & - \end{pmatrix}$$

(b) Step 2

$$S' = (\theta_4, \theta_3, \theta_5)$$

$$\begin{pmatrix} - & 0.38 & - & - & 0.62 \\ 0.60 & - & - & - & 0.40 \\ - & - & - & - & - \\ - & - & - & - & - \\ \multicolumn{5}{c}{\boxed{0.71 \quad 0.29 \quad - \quad - \quad -}} \end{pmatrix}$$

(c) Step 3

$$S' = (\theta_4, \theta_3, \theta_5, \theta_1, \theta_2)$$

$$\begin{pmatrix} - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \end{pmatrix}$$

(d) Step 4

Figure 6: Example of sampling a permutation from Edge Histogram Matrix. Circled areas are used to sample the class of the next position

   (b) Normalize the rows of EHM
   (c) Sample next class using the row of EHM that correspond to the class sampled in the previous position.
3. If the list is not finished, go to step 2.
4. Obtain the final list.

In order to explain it better, we illustrate step by step in Figure 6 how EHBSA generates a new solution based on the EHM shown in Figure 5.

Step 1 (Figure 6(a)): Let consider that θ_4 is selected in the first position. The row that correspond to θ_4 is used to sample the class in position 2.

Step 2 (Figure 6(b)): Let consider that θ_3 is sampled. In the EHM of Step 2 we discard the row and column that correspond to θ_4 and we normalize the rows of EHM. The row that correspond to θ_3 is used to sample the class in position 3.

Step 3 (Figure 6(c)): Let consider that θ_5 is sampled. Again we actualize EHM discarding the row and column that correspond to θ_3 and normalizing the rows, and the row that correspond to θ_5 is used to sample the class in position 4.

Step 4 (Figure 6(d)): Let consider that θ_1 is sampled. As only θ_2 is left we sample it at the last position and we obtain the new solution: $S' = (\theta_4, \theta_3, \theta_5, \theta_1, \theta_2)$.

3.2. Evaluation of samples

In order to select the best samples in EHBSA, we evaluate each sample as follows: in a validation process, for each binary sub-problem the number of well classified instances is calculated. Thus, given a permutation sample, its fitness is the sum of the number of instances well classified in each binary sub-problem. In Figure 7 we illustrate how two individuals are evaluated in a 4 class problem. In the left side it is shown the number of well classified instances for each sub-problem in the validation phase; in the right side two graph samples are shown and how their fitnesses are obtained: summing the number of well classified instances in those sub-problems that are taken into account in the samples.

4. Experiments

In this section we explain the experimental setup of the empirical study we have carried out in order to analyse the performance of DUCG method. We have compared DUCG with several state-of-the-art methods and discuss the obtained results.

4.1. Datasets

In order to evaluate the performance of the proposed approach 27 datasets have been selected from the UCI repository [10]. Table 1 summarizes their properties. In order to complete the information, Table 2 shows the number of instances per class in each database. For the databases with more than 10 classes, in the column denoted as "Mean rest" the mean number of instances of the remaining classes is indicated. Moreover, the last two columns show the mean number of instances and the standard deviation per class.

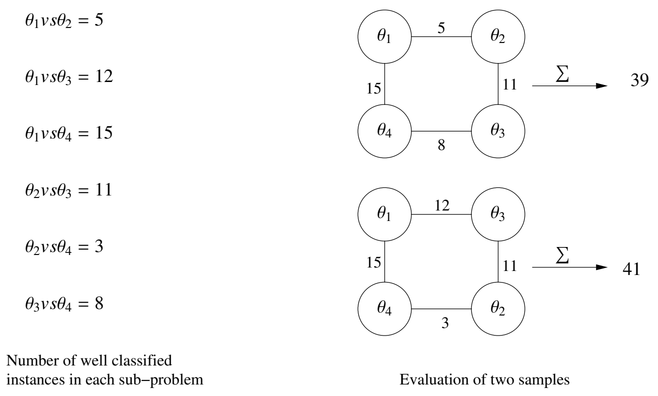

$\theta_1 vs \theta_2 = 5$
$\theta_1 vs \theta_3 = 12$
$\theta_1 vs \theta_4 = 15$
$\theta_2 vs \theta_3 = 11$
$\theta_2 vs \theta_4 = 3$
$\theta_3 vs \theta_4 = 8$
Number of well classified instances in each sub–problem
θ_1
θ_2
θ_3
θ_4
5
15
11
8
Σ
39
12
15
11
3
Σ
41
Evaluation of two samples

Figure 7: Example of the evaluation of samples

4.2. *Base Classifiers*

To carry out the experiments, we have used 4 well known supervised classification algorithms from a software package for Machine Learning called WEKA [19].

- J48 (C4.5 clone)[32], decision tree algorithm. It makes a post-pruning phase, based on error based pruning algorithm.
- SMO (SVM clone)[29], kernel methods. It creates a hyperplane where the categories are divided by a clear gap that is as wide as possible.
- JRip (Ripper clone)[7], rule induction classifier. It builds a rule-set by repeatedly adding rules to an empty rule-set until all positive examples are covered.
- Multilayer Perceptron[34], an artificial neural network. It is a feedforward network of neurons which maps input vectors to output vectors.

In recent reviews, [13] and [17] show that the performance of the different Class Binarization strategies varies depending on the base classifier. Viewing that, in order to give a real perspective, we have selected classifiers with different approaches. As we treat the classifiers as black boxes we have used the default parameters of the classifiers.

4.3. *Strategies summarized*

In this sub-section we briefly describe the Class Binarization strategies that are used for the comparison.
State-of-the-art methods:

- One-vs-All (OVA): Each sub-problem compares one class with the rest of classes. The class with the highest confidence level is selected.
- One-vs-One (OVO) [12, 11]: Each sub-problem compares two classes between them, ignoring the rest. And the majority vote is used to take the final decision.
- Decision Directed Acyclic Graph (DDAG) [30]: The DDAG is equivalent to operating on a list. A list is initialized with all the classes. In each step a classifier discriminates between two classes selected from the list, and the class which is not selected is eliminated. The DDAG terminates when only one class remains in the list.
- All-And-One (A&O) [16, 25]: Combination of OVA and OVO. First OVA is applied and the two classes with the highest confidence level are selected. A classifier that discriminates between the selected classes is built and the result of the classifier is the final decision.

Our proposals:

Domain	#Instances	#Attrib	#Classes
Car	1728	6	4
Vehicle	846	18	4
Annealing	798	38	5
Gesture	9873	32	5
Nursery	12960	8	5
Page-blocks	5473	10	5
Autouniv	25000	45	6
Dermatology	366	33	6
Flare	1066	11	6
Glass	214	9	6
Satimage	6435	36	6
Winequality Red	1599	10	6
Image Segmentation	2310	19	7
Shuttle	58000	9	7
Winequality White	4898	10	7
Zoo	101	16	7
Ecoli	336	7	8
Optdigits	5620	64	10
Pendigits	10992	16	10
Yeast	1484	8	10
Pokerhand	25010	10	10
Vowel	990	12	11
Arrhythmia	452	279	13
Chess	28056	6	18
Soybean	683	35	19
Letters	20000	16	26
Abalone	4177	8	28

Table 1: The main characteristics of the 27 databases

- DUCG-Rand: Algorithm proposed in Section 3 where the order of the list is decided randomly.
- DUCG-EHBSA: Algorithm proposed in Section 3 where the order of the list is decided with EHBSA.

To see the performance of the proposed approach we have compared our algorithm with other state-of-the-art methods. Moreover, in our method we propose to use EHBSA in order to select the proper order of the classes; however, we have considered suitable to compare it with DUCG-Rand to remark the obtained benefits of the used strategy.

4.4. Performance measures

Several performance measures can be found in the literature. Due to its simplicity, the Classification Rate is the most commonly used metric for calculating the accuracy of classifiers. However, Ben-David [4] showed that several hits can be attributed to chance, in order to compensate the random hits he proposed to use Cohen's Kappa metric [6]. Following Galar's et. al overview [13] both metrics are used in this paper.

- Classification rate: Also is called accuracy. Among all the classified instances, it calculates the proportion of well classified ones.
- Cohen's Kappa [6]: This metric tries to calculate the portion of hits that can be attributed to the classifier itself and are not obtained by chance.

  $$kappa = \frac{P_0 - P_c}{1 - P_c} \tag{1}$$

  where P_0 is the total agreement probability and P_c is the agreement probability that is due to chance.

  Cohen's Kappa also can be easily illustrated through use of a confusion matrix, and Equation 1 is equivalent to this one:

  $$kappa = \frac{n \sum_{i=1}^{K} h_{ii} - \sum_{i=1}^{K} T_{ri} T_{ci}}{n^2 - \sum_{i=1}^{K} T_{ri} T_{ci}} \tag{2}$$

Domain	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	Mean rest	Mean	σ
Car	1210	384	69	65								432.0	±539.8
Vehicle	218	217	212	199								211.5	±8.7
Annealing	608	88	60	34	8							159.6	±252.4
Gesture	2950	2741	2097	1087	998							1974.6	±907.7
Nursery	432	426	405	32	1							259.2	±222.1
Page-blocks	4913	329	115	88	28							1094.6	±2137.6
Autouniv	8345	7981	3309	1987	1813	1565						4166.7	±3156.0
Dermatology	112	72	61	52	49	20						61.0	±30.4
Flare	331	239	211	147	95	43						177.7	±104.2
Glass	76	70	29	17	13	9						35.7	±29.7
Satimage	1533	1508	1358	707	703	626						1072.5	±436.5
Winequality Red	681	638	199	53	18	10						266.5	±312.3
Image Segmentation	330	330	330	330	330	330	330					330.0	±0.0
Shuttle	45580	9004	3191	159	46	11	9					8285.7	±16772.2
Winequality White	2198	1457	880	175	163	20	5					699.7	±852.3
Zoo	41	20	13	10	8	5	4					14.4	±12.9
Ecoli	143	77	52	35	20	5	2	2				42.0	±48.7
Optdigits	572	571	568	566	562	558	558	557	554	554		562.0	±6.8
Pendigits	1144	1144	1143	1143	1142	1056	1055	1055	1055	1055		1099.2	±46.4
Yeast	463	429	244	163	51	44	37	30	20	5		148.6	±173.5
Pokerhand	10599	12493	1206	513	93	54	36	6	5	5		2501	±4802.7
Vowel	90	90	90	90	90	90	90	90	90	90	90	90	±0.0
Arrhythmia	245	50	44	25	22	15	15	13	9	5	3	34.8	±64.9
Chess	4553	4194	3597	2854	2796	2166	1985	1712	1433	683	260.38	1558.7	± 1503.2
Soybean	92	91	91	88	44	44	20	20	20	20	17	35.9	±30.2
Letters	813	805	803	796	792	789	787	786	783	783	753.94	769.2	±23.2
Abalone	689	634	568	487	391	267	259	203	115	103	19.71	150.0	±214.8

Table 2: Class distribution, mean and standard deviation of the 27 databases

where n is the number of examples, K is the number of class labels, h_{ii} is the number of true positives for each class (elements of the main diagonal) and T_{ri} and T_{ci} are the total sum of the i-th row and column, respectively ($T_{ri} = \sum_{j=1}^{m} h_{ij}$, $T_{ci} = \sum_{j=1}^{m} h_{ji}$).

Cohen's Kappa ranges from -1 (total disagreement) through 0 (random classification) to 1 (perfect agreement). However, most classifiers do at least as good as random, so by definition they score kappa higher than 0.

4.5. Experimental setup

In the experimental phase 5x2 fold cross-validation has been used. As the proposed approach needs the best order to be fixed, a pre-process step is applied in each fold. It consists on a five times repeated hold-out in which 70% of the cases are used as validation and the remaining 30% are used to tune the order candidates.

4.6. Obtained results

In this sub-section the accuracy and Cohen's Kappa results obtained with the different base classifiers are shown. In order to illustrate better the obtained results, they are shown in tables where the databases are ordered by the number of classes. Moreover, each table is divided into 3 sections: in the first section the results are shown, in the second section the average results and average ranking for each method are shown and in the third section are shown the average results and average ranking for each method only considering the 10 databases with more than 9 classes. In all these tables we will show that OVO and DUCG-EHBSA obtain the most promising results.

Tables 3 and 4 show the accuracies and kappa results obtained with SVM. The results follow similar pattern in both tables and it can be seen that DUCG-EHBSA gets the best result in the majority of the cases: 15 in Table 3 (accuracy) and 12 in Table 4 (kappa). Furthermore, in both cases DUCG-EHBSA achieves the best mean and rank. It can be seen also that DUCG-EHBSA obtains interesting results in databases with high number of classes; it obtains the best mean, rank and the best results in 6 of those databases. On the other hand, OVA receives the worst results. The reason of this fact is that for some instances all the outputs are negative, with 1.0 confidence level, hence all classes are tied, and in this case the most represented class is returned.

Tables 5 and 6 show the accuracies and kappa results achieved with Ripper. Taking into account only the accuracy (Table 5), it can be observed that OVO gets the best results: it obtains the best result in the majority of the cases (in 12 databases) and also it obtains the best mean and rank values. Moreover, it can be seen that in the databases with more

Table 3: Classification accuracies of different methods using SVM

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	73.218	83.484	83.264	82.280	83.299	**83.484**
Vehicle	52.931	71.749	71.797	71.820	71.655	**71.891**
Annealing	83.408	84.009	83.831	**85.011**	83.987	83.942
Gesture	29.879	45.318	45.330	45.293	45.224	**45.678**
Nursery	78.244	**90.909**	**90.909**	90.253	**90.909**	**90.909**
PageBlocks	91.891	93.506	**93.689**	92.721	93.674	93.528
Autouniv	47.960	53.754	54.648	52.601	54.365	**55.454**
Dermatology	95.519	97.268	97.268	**97.486**	97.268	97.268
Flare	38.574	60.525	60.619	60.469	60.563	**60.619**
Glass	44.673	52.430	52.336	**52.897**	52.710	**52.897**
Satimage	73.445	**86.692**	86.670	85.949	86.667	86.645
WineRed	45.641	57.386	57.411	**57.448**	57.373	57.386
ImgSeg	77.680	92.823	**92.831**	92.571	92.814	92.814
Shuttlle	84.081	97.189	97.087	96.744	97.101	**97.203**
WineWhite	47.162	**51.940**	51.935	**51.940**	**51.940**	**51.940**
Zoo	90.297	**93.663**	92.277	92.871	92.871	92.871
Ecoli	65.357	81.488	**81.845**	81.190	81.726	81.667
OptDig	92.285	97.972	97.890	97.431	97.886	**98.000**
Pendig	86.619	97.698	97.575	96.090	97.706	**97.775**
Yeast	38.598	55.849	55.822	55.836	**55.970**	55.889
Pokerhand	**49.952**	**49.952**	**49.952**	**49.952**	**49.952**	**49.952**
Vowel	14.505	67.354	67.535	66.182	67.495	**68.101**
Arrhythmia	65.310	67.345	66.593	**68.274**	66.858	66.770
Chess	16.349	**35.086**	34.376	33.960	34.472	35.075
Soybean	91.567	92.152	91.654	**93.353**	92.328	92.592
Letters	32.062	82.328	81.867	80.507	82.035	**82.384**
Abalone	16.495	25.142	25.229	25.085	25.186	**25.238**
Mean	60.137	72.778	72.676	72.452	72.742	**72.888**
Rank	5.91	2.85	3.22	3.72	3.09	**2.20**
Mean>9Class	50.374	67.088	66.849	66.667	66.989	**67.178**
Rank>9Class	5.75	2.85	3.75	3.95	2.85	**1.85**

Table 4: Cohen's Kappa results of different methods using SVM. When "*" appears before a database name, it indicates that for this database the accuracy of OVO is greater than the accuracy of DUCG-EHBSA, but the kappa result is worse.

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
*Car	0.20750	0.60973	0.61059	0.57835	**0.61093**	0.60977
Vehicle	0.38171	0.62425	0.62475	0.62525	0.62290	**0.62605**
Annealing	0.43214	0.47855	0.47503	**0.52090**	0.47813	0.47722
Gesture	0.00000	0.23477	0.23585	0.23436	0.23440	**0.24197**
Nursery	0.67687	**0.86603**	**0.86603**	0.85559	**0.86603**	**0.86603**
PageBlocks	0.33417	0.55207	0.57426	0.44683	**0.57429**	0.56889
Autouniv	0.22310	0.34002	0.36181	0.31140	0.35407	**0.37706**
Dermatology	0.94345	0.96574	0.96574	**0.96848**	0.96574	0.96574
Flare	0.12802	0.48630	**0.48790**	0.48503	0.48714	0.48761
Glass	0.21449	0.31885	0.32023	0.32736	0.32520	**0.32865**
Satimage	0.66162	**0.83498**	0.83479	0.82540	0.83474	0.83442
WineRed	0.06683	0.27504	0.27571	**0.27605**	0.27483	0.27504
ImgSeg	0.74025	0.91624	**0.91634**	0.91331	0.91614	0.91614
Shuttlle	0.37492	0.92047	0.91783	0.90661	0.91820	**0.92090**
WineWhite	0.05774	0.18940	0.18936	0.18940	**0.18941**	0.18940
Zoo	0.86738	**0.91550**	0.89754	0.90525	0.90552	0.90536
Ecoli	0.46162	0.73910	**0.74455**	0.73418	0.74263	0.74168
OptDig	0.91429	0.97746	0.97655	0.97145	0.97651	**0.97777**
Pendig	0.85127	0.97442	0.97305	0.95654	0.97450	**0.97527**
Yeast	0.13314	0.41504	0.41609	0.41408	**0.41762**	0.41640
Pokerhand	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**	**0.00000**
Vowel	0.07474	0.64139	0.64325	0.62871	0.64283	**0.64947**
Arrhythmia	0.36811	0.46341	0.46259	0.46308	**0.46429**	0.46234
*Chess	0.00177	0.26695	0.26100	0.25170	0.26176	**0.26727**
Soybean	0.90733	0.91381	0.90833	**0.92703**	0.91578	0.91869
Letters	0.29371	0.81620	0.81140	0.79726	0.81315	**0.81678**
Abalone	0.00000	0.13315	**0.13464**	0.13198	0.13414	0.13453
Mean	0.38208	0.58774	0.58834	0.57947	0.58892	**0.59076**
Rank	5.91	3.19	2.87	3.94	2.74	**2.35**
Mean>9Class	0.35444	0.56018	0.55869	0.55418	0.56006	**0.56185**
Rank>9Class	5.75	3.05	3.35	4.25	2.65	**1.95**

classes OVO obtains the best results. However, in Table 6 the results are not so differential. This time, OVO gets the best result in 8 databases and is nearly followed by DUCG-EHBSA which obtains the best results in 6. Furthermore, the means of both methods are similar, slightly better the OVO's one. It can be observed that the rank is in favour of DUCG-EHBSA. OVO continues having the the best mean for databases with more classes, but the rank is equal for OVO and DUCG-EHBSA.

Table 5: Classification accuracies of different methods using Ripper

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	92.940	94.005	93.935	**94.363**	93.900	93.808
Vehicle	**68.463**	67.069	66.950	67.470	67.234	67.305
Annealing	93.742	**94.165**	93.541	93.363	93.697	93.808
Gesture	46.871	**51.417**	50.815	46.703	51.123	50.990
Nursery	**98.744**	97.802	97.785	97.744	97.779	97.798
PageBlocks	96.722	96.726	96.653	96.722	96.715	**96.781**
Autouniv	63.682	65.687	**65.697**	65.690	65.648	65.654
Dermatology	90.328	94.645	**94.863**	94.098	94.754	94.536
Flare	56.323	59.456	59.362	**59.756**	59.287	59.362
Glass	60.467	**65.140**	64.486	61.589	**65.140**	65.047
Satimage	85.246	**86.782**	86.151	85.815	86.427	86.567
WineRed	**57.674**	57.123	56.748	57.486	56.898	56.923
ImgSeg	93.671	94.251	94.286	94.398	94.390	**94.554**
Shuttlle	**99.957**	99.951	99.948	99.950	99.951	99.951
WineWhite	53.018	**54.447**	53.859	53.744	54.087	54.390
Zoo	**90.693**	87.723	88.317	90.297	88.515	88.713
Ecoli	77.738	81.250	81.071	80.179	80.714	**81.369**
OptDig	89.349	**92.865**	90.068	91.085	91.327	91.278
Pendig	94.256	**96.021**	95.093	94.914	95.482	95.639
Yeast	54.299	**56.685**	56.199	55.849	56.442	56.321
Pokerhand	55.212	55.640	55.750	55.701	55.764	**56.122**
Vowel	58.404	**66.747**	63.717	61.576	65.293	66.101
Arrhythmia	65.929	67.168	65.088	**68.009**	66.372	66.770
Chess	41.433	**63.763**	60.769	47.411	62.017	62.824
Soybean	88.404	90.249	88.960	89.693	89.370	**90.307**
Letters	82.283	**88.816**	83.745	83.866	86.032	86.391
Abalone	18.937	**26.579**	25.765	21.350	26.215	26.411
Mean	73.140	**76.006**	75.171	74.401	75.577	75.767
Rank	4.80	**2.20**	4.20	3.87	3.39	2.54
Mean>9Class	64.851	**70.453**	68.515	66.945	69.431	69.816
Rank>9Class	5.90	**1.60**	4.40	4.10	2.90	2.10

Tables 7 and 8 show the accuracies and kappa results obtained with C4.5. The patterns of these tables are similar to those obtained with Ripper. In Table 7 the results are in favour of OVO. It gets the best results in 15 databases. Furthermore, it can be seen that OVO obtains the best results specially with databases with more classes. Nevertheless, as in Ripper, the results in kappa are slightly different. Although OVO continues obtaining the best mean, the difference is lower and DUCG-EHBSA's one is close to it. Moreover, DUCG-EHBSA acquires the best rank and the superiority of OVO in databases with more classes is decreased since the mean difference is low and DUCG-EHBSA achieves better rank.

Finally Tables 9 and 10 show the accuracies and kappa results obtained with Multilayer Perceptron. In Table 9 OVA gets the best accuracy in the majority of the cases and is closely followed by OVO. OVO achieves the best mean and rank, but these values are nearly from those obtained by DUCG-EHBSA. In this table it can be appreciated that our proposed approach DUCG-EHBSA achieves the best results for the databases with more classes. On the other hand, in Table 10, the best kappa results are more distributed. In this case, it is DUCG-EHBSA which obtains the best mean and rank. Furthermore, in this time also, DUCG-EHBSA achieves the best results for the databases with more classes.

Summarizing the results obtained from this analysis we conclude that OVO and DUCG-EHBSA are the most robust approaches, OVO performs better with C4.5 and Ripper whereas DUCG-EHBSA performs better with SVM and Multilayer Perceptron. In fact, Multilayer Perceptron is the base classifier that obtains the best results among the base classifiers. Besides, it can be seen that when kappa is considered DUCG-EHBSA achieves interesting results. We want to emphasize also the results obtained by DUCG-RAND, where in most of the cases it obtains better mean and rank than OVA, DDAG and A&O. In addition to this, it can be seen that in almost all the methods there is a considerable difference between the mean of OVO, DUCG-EHBSA and DUCG-RAND, and the remaining methods,

Table 6: Cohen's Kappa results of different methods using Ripper . When "*" appears before a database name, it indicates that for this database the accuracy of OVO is greater than the accuracy of DUCG-EHBSA, but the kappa result is worse.

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	0.84428	0.87061	0.86975	**0.87794**	0.86886	0.86659
Vehicle	**0.57928**	0.56166	0.55944	0.56586	0.56316	0.56438
Annealing	0.83758	**0.84995**	0.83541	0.83287	0.83920	0.84179
*Gesture	0.26626	0.34259	0.34310	0.27525	**0.34755**	0.34449
Nursery	**0.98158**	0.96779	0.96754	0.96691	0.96745	0.96772
PageBlocks	0.81750	0.82641	0.82491	0.82504	0.82793	**0.83050**
*Autouniv	0.50419	0.52775	**0.53151**	0.52584	0.53074	0.52839
Dermatology	0.87911	0.93296	**0.93572**	0.92613	0.93439	0.93163
Flare	0.43428	0.47710	0.47744	**0.47797**	0.47658	0.47707
*Glass	0.44605	0.50776	0.50711	0.45663	**0.51537**	0.51387
Satimage	0.81837	**0.83639**	0.82898	0.82500	0.83232	0.83400
WineRed	**0.31107**	0.30165	0.30153	0.29717	0.30514	0.29993
ImgSeg	0.92613	0.93292	0.93331	0.93462	0.93453	**0.93645**
Shuttlle	**0.99878**	0.99861	0.99853	0.99858	0.99862	0.99861
*WineWhite	0.23372	0.27723	**0.27842**	0.24210	0.27788	0.27794
Zoo	**0.87685**	0.83801	0.84572	0.87267	0.84842	0.85134
Ecoli	0.69311	0.73605	0.73502	0.72473	0.72985	**0.73911**
OptDig	0.88166	**0.92072**	0.88963	0.90094	0.90363	0.90307
Pendig	0.93615	**0.95577**	0.94546	0.94348	0.94979	0.95153
Yeast	0.39506	**0.43398**	0.43129	0.41293	0.43227	0.43060
Pokerhand	0.14100	0.16611	0.17137	0.16690	0.16965	**0.17656**
Vowel	0.54131	**0.63479**	0.60106	0.57695	0.61848	0.62730
*Arrhythmia	0.43117	0.45464	0.45904	**0.49025**	0.47652	0.48317
Chess	0.33332	**0.59368**	0.56207	0.40113	0.57557	0.58391
Soybean	0.87223	0.89277	0.87870	0.88647	0.88318	**0.89344**
Letters	0.81576	**0.88368**	0.83094	0.83222	0.85472	0.85846
*Abalone	0.04142	0.16163	0.15930	0.07362	0.16144	**0.16263**
Mean	0.62360	**0.66234**	0.65564	0.64112	0.66012	0.66202
Rank	5.0	2.76	3.74	4.11	2.93	**2.46**
Mean>9Class	0.53891	**0.60978**	0.59289	0.56849	0.60252	0.60707
Rank>9Class	6.00	**2.00**	4.40	4.10	2.90	**2.00**

Table 7: Classification accuracies of different methods using C4.5

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	95.972	95.914	95.799	**96.215**	95.787	95.741
Vehicle	68.251	68.251	67.825	**68.983**	68.203	68.298
Annealing	92.116	92.227	91.960	**92.272**	91.893	92.183
Gesture	49.618	**53.202**	51.105	50.609	51.881	52.302
Nursery	**98.647**	98.622	98.608	98.603	98.608	98.622
PageBlocks	96.642	96.824	96.810	96.645	96.835	**96.850**
Autouniv	61.777	**64.727**	64.379	61.370	64.499	64.519
Dermatology	91.530	**95.574**	95.355	94.262	95.301	95.410
Flare	56.023	59.962	59.812	**60.225**	59.606	59.887
Glass	60.374	**63.084**	61.682	60.748	61.589	62.710
Satimage	83.708	**85.946**	85.442	84.525	85.678	85.803
WineRed	**58.649**	57.486	57.198	57.899	57.298	57.286
ImgSeg	94.251	95.030	94.857	94.563	95.100	**95.299**
Shuttlle	99.949	99.944	99.945	**99.960**	99.946	99.948
WineWhite	53.973	**54.924**	53.748	54.904	54.079	54.892
Zoo	90.693	90.297	90.693	**92.673**	90.297	90.693
Ecoli	78.571	**81.726**	81.190	79.048	81.548	81.429
OptDig	87.434	**92.295**	89.288	89.356	90.669	90.751
Pendig	94.039	**95.941**	94.985	94.649	95.298	95.486
Yeast	55.418	**56.267**	55.755	55.984	56.253	56.132
Pokerhand	49.730	49.895	50.138	49.483	50.026	**50.625**
Vowel	66.121	**71.434**	67.556	68.768	69.939	70.141
Arrhythmia	63.496	**65.885**	62.788	61.460	65.133	64.867
Chess	54.569	**63.669**	61.008	58.308	62.635	62.735
Soybean	86.559	**90.893**	89.136	88.316	90.015	90.571
Letters	83.181	**89.002**	83.595	84.398	86.199	86.587
Abalone	18.793	24.946	24.228	20.062	24.870	**25.310**
Mean	73.707	**76.073**	74.996	74.603	75.525	75.744
Rank	4.87	**2.02**	4.31	3.89	3.41	2.43
Mean>9Class	65.934	**70.023**	67.848	67.078	69.104	69.321
Rank>9Class	5.70	**1.40**	4.30	4.80	2.80	2.00

Table 8: Cohen's Kappa results of different methods using C4.5. When "*" appears before a database name, it indicates that for this database the accuracy of OVO is greater than the accuracy of DUCG-EHBSA, but the kappa result is worse.

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	0.91185	0.91117	0.90882	**0.91745**	0.90861	0.90752
Vehicle	0.57654	0.57854	0.57228	**0.58756**	0.57741	0.57822
Annealing	0.79060	0.79565	0.78992	**0.79622**	0.78915	0.79527
Gesture	**0.48860**	0.37119	0.35682	0.34830	0.36608	0.36912
Nursery	**0.98017**	0.97981	0.97961	0.97955	0.97961	0.97981
PageBlocks	0.81941	0.82825	0.82905	0.82270	0.82986	**0.83023**
*Autouniv	0.48259	0.51600	0.51736	0.47977	**0.51882**	0.51740
Dermatology	0.89396	**0.94451**	0.94182	0.92808	0.94118	0.94251
Flare	0.42048	**0.48183**	0.48078	0.48044	0.47794	0.48135
*Glass	0.46057	0.48535	0.47544	0.46890	0.47482	**0.48558**
Satimage	0.79860	**0.82593**	0.82019	0.80893	0.82305	0.82448
WineRed	**0.33021**	0.30979	0.31267	0.31821	0.31498	0.30888
ImgSeg	0.93291	0.94201	0.93998	0.93655	0.94281	**0.94514**
Shuttlle	0.99856	0.99842	0.99846	**0.99887**	0.99849	0.99853
*WineWhite	0.27633	0.29874	0.29432	**0.30170**	0.29791	0.30038
Zoo	0.87754	0.87118	0.87658	**0.90327**	0.87137	0.87661
Ecoli	0.70339	**0.74637**	0.74100	0.71067	0.74558	0.74386
OptDig	0.86037	**0.91439**	0.88097	0.88172	0.89631	0.89722
Pendig	0.93375	**0.95488**	0.94427	0.94053	0.94775	0.94983
Yeast	0.41296	0.42735	0.42423	0.42209	**0.42880**	0.42703
Pokerhand	0.06596	0.05349	0.06139	0.06021	0.05881	**0.06863**
Vowel	0.62748	**0.68599**	0.64299	0.65629	0.66932	0.67158
*Arrhythmia	0.45228	0.42685	0.43234	0.42938	**0.45915**	0.45592
Chess	0.33497	**0.59191**	0.56382	0.53112	0.58155	0.58226
Soybean	0.85171	**0.89983**	0.88058	0.87131	0.89017	0.89634
Letters	0.82508	**0.88561**	0.82938	0.83773	0.85646	0.86050
Abalone	0.04087	0.13551	0.13541	0.05793	0.13849	**0.14335**
Mean	0.63510	**0.66150**	0.65298	0.64724	0.65868	0.66065
Rank	4.59	2.65	4.13	3.93	3.31	**2.39**
Mean>9Class	0.54054	**0.59758**	0.57954	0.56883	0.59268	0.59527
Rank>9Class	5.30	2.35	4.20	4.60	2.65	**1.90**

Table 9: Classification accuracies of different methods using Multilayer Perceptron

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	95.162	**95.868**	95.845	95.660	95.764	95.856
Vehicle	**80.449**	79.196	79.574	80.236	79.314	79.551
Annealing	98.040	**98.151**	98.129	98.062	**98.151**	**98.151**
Gesture	50.793	**51.255**	50.511	50.957	51.062	50.888
Nursery	98.119	99.466	99.469	99.255	99.468	**99.474**
PageBlocks	96.079	96.397	96.371	**96.401**	96.357	96.357
Autouniv	**61.338**	60.026	58.811	61.191	59.318	59.227
Dermatology	96.066	**96.995**	**96.995**	**96.995**	**96.995**	**96.995**
Flare	**59.568**	58.780	58.856	58.949	58.630	58.874
Glass	**65.234**	64.579	64.112	64.206	64.206	63.738
Satimage	89.330	89.551	89.483	**89.650**	89.532	89.629
WineRed	**58.487**	57.836	57.674	58.186	57.799	57.736
ImgSeg	96.052	96.554	96.563	96.433	**96.623**	96.528
Shuttlle	99.647	99.771	99.766	99.705	99.764	**99.778**
WineWhite	**54.153**	53.699	53.018	53.173	53.499	53.687
Zoo	93.663	94.059	93.465	93.069	93.663	**94.455**
Ecoli	**86.190**	85.179	84.762	85.774	85.060	85.119
OptDig	97.890	97.886	97.801	97.954	97.886	**97.989**
Pendig	95.213	**99.010**	98.956	95.122	98.983	98.997
Yeast	**58.693**	57.480	57.264	57.642	57.224	57.453
Pokerhand	**53.525**	52.300	52.450	52.457	52.385	52.830
Vowel	85.071	88.646	88.505	85.212	**89.091**	89.071
Arrhythmia	65.575	**68.319**	67.168	67.434	67.965	68.230
Chess	58.926	**62.546**	60.716	60.979	61.459	62.054
Soybean	**92.943**	91.332	91.157	91.654	91.742	91.567
Letters	86.467	**93.084**	91.865	86.750	92.710	92.828
Abalone	26.052	26.220	25.564	**26.517**	25.746	26.004
Mean	77.731	**78.303**	77.957	77.764	78.163	78.262
Rank	3.76	**2.76**	4.48	3.43	3.63	2.94
Mean>9Class	72.035	73.682	73.145	72.172	73.519	**73.702**
Rank>9Class	3.80	2.75	4.90	3.50	3.55	**2.50**

Table 10: Cohen's Kappa results of different methods using Multilayer Perceptron. When "*" appears before a database name, it indicates that for this database the accuracy of OVO is greater than the accuracy of DUCG-EHBSA, but the kappa result is worse.

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	0.89512	**0.91024**	0.90995	0.90596	0.90827	0.91005
Vehicle	**0.73917**	0.72220	0.72711	0.73603	0.72365	0.72679
Annealing	0.95064	**0.95330**	0.95279	0.95123	0.95329	0.95329
*Gesture	0.34651	0.34724	0.34391	0.35040	**0.35104**	0.34815
Nursery	0.97232	0.99218	0.99222	0.98909	0.99220	**0.99229**
PageBlocks	0.76714	0.79907	0.79928	**0.80122**	0.79885	0.79875
Autouniv	**0.47417**	0.45137	0.44608	0.47249	0.45169	0.44872
Dermatology	0.95070	**0.96232**	**0.96232**	**0.96232**	**0.96232**	**0.96232**
Flare	**0.48135**	0.47356	0.47565	0.47647	0.47222	0.47541
Glass	**0.51690**	0.51257	0.51020	0.51028	0.51056	0.50399
Satimage	0.86804	0.87074	0.87003	**0.87207**	0.87066	0.87175
*WineRed	0.32796	0.32486	0.32836	**0.33254**	0.32897	0.32581
ImgSeg	0.95392	0.95979	0.95988	0.95837	**0.96059**	0.95948
Shuttlle	0.99004	0.99353	0.99338	0.99166	0.99335	**0.99374**
*WineWhite	**0.28690**	0.26485	0.26412	0.26470	0.26669	0.26579
Zoo	0.91563	0.92105	0.91347	0.90796	0.91590	**0.92600**
Ecoli	**0.80797**	0.79426	0.78900	0.80296	0.79279	0.79364
OptDig	0.97655	0.97651	0.97556	0.97726	0.97651	**0.97766**
Pendig	0.94679	**0.98900**	0.98839	0.94578	0.98870	0.98886
*Yeast	**0.46408**	0.44286	0.44272	0.44795	0.44117	0.44390
Pokerhand	**0.11204**	0.08084	0.09086	0.08476	0.08698	0.09389
Vowel	0.83571	0.87508	0.87347	0.83725	**0.87992**	0.87971
*Arrhythmia	0.43798	0.48585	0.48418	0.48592	0.48635	**0.49054**
Chess	0.53974	**0.57988**	0.56085	0.56284	0.56907	0.57516
Soybean	**0.92257**	0.90481	0.90292	0.90842	0.90935	0.90744
Letters	0.85924	**0.92807**	0.91539	0.86219	0.92418	0.92541
*Abalone	0.15764	0.16152	0.15943	**0.16444**	0.15943	0.16176
Mean	0.68507	0.69176	0.69006	0.68750	0.69166	**0.69260**
Rank	4.0	3.28	4.20	3.41	3.28	**2.83**
Mean>9Class	0.62523	0.64244	0.63938	0.62768	0.64217	**0.64443**
Rank>9Class	4.10	3.25	4.65	3.60	3.30	**2.10**

and this difference is increased when the databases with 10 or more classes are considered.

However, we can not obtain any meaningful conclusion without using a statistical test. Hence, in the next subsection, we carry out an statistical analysis in order to find whether significant differences among the results obtained exist or not.

4.6.1. Statistical analysis

As we have several methods to compare, according to García et al. [15], we have used the Iman-Davenport test to detect statistical differences among the different strategies. If the difference exists, we apply the Shaffer post-hoc test in order to find out which algorithms are distinctive among them. We show the most relevant p-values obtained in the pairwise comparisons in tables, where "+" symbol implies that the first algorithm is statistically better than the confronting one, whereas "=" means that there are not significant differences between them.

With respect to SVM, the results of the statistical analysis reject the null hypothesis that all the methods are equivalent, since the p-values returned by the Iman-Davenport test are lower than our α-value (0.1) for both performance measures. In Table 11 we show the most relevant p-values obtained with Shaffer post-hoc test. In both cases all the strategies outperform significantly OVA, mainly because OVA obtains the worst result in all the databases. This fact makes to be more difficult to find more statistical differences since the p-value is re-adjusted after each pairwise comparison in Shaffer post-hoc test. However, DUCG-EHBSA also outperforms A&O in both tables. Viewing these results we consider that DUCG-EHBSA is the most suitable method for SVM.

Considering Ripper, the Iman-Davenport test returns p-values lowers than 0.0001 for both cases, so we execute the Shafer post-hoc test. The obtained p-values can be seen in Table 12. The accuracy results show that OVO and DUCG-EHBSA outperform significantly OVA, DDAG and A&O, whereas DUCG-RAND outperforms OVA. The Kappa results are similar since OVO and DUCG-EHBSA get significantly better results than OVA and A&O. Seeing these results we conclude that OVO and DUCG-EHBSA are equivalent between them and they perform better than other approaches for Ripper.

Concerning C4.5, this time again the obtained p-values in Iman-Davenport test are very low, lower than 0.0001.

Table 11: Shaffer test for SVM base classifier

Accuracy	
DUCG-EHBSA vs OVA	**+(5.2E-12)**
OVO vs OVA	**+(1.9E-8)**
DUCG-RAND vs OVA	**+(3.2E-7)**
DDAG vs OVA	**+(1.3E-6)**
A&O vs OVA	**+(1.8E-4)**
DUCG-EHBSA vs A&O	**+(0.0286)**
DUCG-EHBSA vs DDAG	=(0.3183)
DUCG-EHBSA vs DUCG-RAND	=(0.5660)
OVO vs A&O	=(0.6117)
Kappa	
DUCG-EHBSA vs OVA	**+(4.3E-11)**
DUCG-RAND vs OVA	**+(5.0E-9)**
DDAG vs OVA	**+(2.5E-8)**
OVO vs OVA	**+(9.0E-7)**
A&O vs OVA	**+(0.0012)**
DUCG-EHBSA vs A&O	**+(0.0176)**
DUCG-RAND vs A&O	=(0.1265)
DDAG vs A&O	=(0.2443)
DUCG-EHBSA vs OVO	=(0.7119)
OVO vs A&O	=(0.8155)

Table 12: Shaffer test for Ripper base classifier

Accuracy	
OVO vs OVA	**+(5.3E-6)**
DUCG-EHBSA vs OVA	**+(9.1E-5)**
OVO vs DDAG	**+(8.6E-4)**
OVO vs A&O	**+(0.0106)**
DUCG-EHBSA vs DDAG	**+(0.0106)**
DUCG-RAND vs OVA	**+(0.0571)**
DUCG-EHBSA vs A&O	**+(0.0618)**
OVO vs DUCG-RAND	=(0.1395)
A&O vs OVA A&O	=(0.4829)
DUCG-EHBSA vs DUCG-RAND	=(0.5660)
DUCG-RAND vs DDAG	=(0.5660)
DDAG vs OVA	=(0.9780)
Kappa	
DUCG-EHBSA vs OVA	**+(9.4E-6)**
OVO vs OVA	**+(1.1E-4)**
DUCG-RAND vs OVA	**+(4.6E-4)**
DUCG-EHBSA vs A&O	**+(0.0121)**
OVO vs A&O	**+(0.0793)**
DUCG-EHBSA vs DDAG	=(0.1209)
DDAG vs OVA	=(0.1209)
DUCG-RAND vs A&O	=(0.1395)
OVO vs DDAG	=(0.3773)
A&O vs OVA	=(0.4851)
DUCG-RAND vs DDAG	=(0.4851)

In Table 13 we show the results obtained with Shaffer pos-hoc test. The p-values obtained in accuracy indicate that OVO outperforms OVA, DDAG, A&O and DUCG-RAND. DUCG-EHBSA also obtains interesting results since it overcomes OVA, DDAG and A&O. And DUCG-RAND outperforms OVA. The Kappa results, however, show that DUCG-EHBSA continues outperforming OVA, DDAG and A&O, but OVO only obtains significant improvements against OVA and DDAG. Viewing that, we conclude that OVO and DUCG-EHBSA are equivalent and are the most robust strategies.

Finally, we apply the statistical test to the results obtained with Multilayer Perceptron. The Iman Davenport test rejects the null hypothesis of equivalence of accuracy (p-value 0.026), but it does not reject the null hypothesis for kappa (p-value 0.113). We execute Shaffer post-hoc for accuracy and the results are shown in Table 14. Once again, OVO and DUCG-EHBSA perform better than the other approaches. On the other hand, although there are no statistical differences among methods in kappa, the p-value is very low and regarding the mean and rank results we may stress the good behaviour of DUCG-EHBSA.

Viewing all these results, we conclude that the most robust strategies are OVO and DUCG-EHBSA. They show

Table 13: Shaffer test for C4.5 base classifier

Accuracy	
OVO vs OVA	**+(3.2E-7)**
DUCG-EHBSA vsOVA	**+(1.6E-5)**
OVO vs DDAG	**+(6.5E-5)**
DUCG-EHBSA vs DDAG	**+(0.0021)**
OVO vs A&O	**+(0.0024)**
OVO vs DUCG-RAND	**+(0.0406)**
DUCG-EHBSA vs A&O	**+(0.0406)**
DUCG-RAND vs OVA	**+(0.0446)**
DUCG-EHBSA vs DUCG-RAND	=(0.2672)
A&O vs OVA	=(0.3234)
DUCG-RAND vs DDAG	=(0.4068)
Kappa	
DUCG-EHBSA vs OVA	**+(2.3E-4)**
OVO vs OVA	**+(0.0013)**
DUCG-EHBSA vs DDAG	**+(0.0063)**
DUCG-EHBSA vs A&O	**+(0.0254)**
OVO vs DDAG	**+(0.0362)**
DUCG-RAND vs OVA	=(0.1209)
OVO vs A&O	=(0.1209)
DUCG-EHBSA vs DUCG-RAND	=(0.4829)
DUCG-RAND vs DDAG	=(0.7668)

Table 14: Shaffer test for Multilayer Perceptron base classifier

Accuracy	
OVO vs DDAG	**+(0.0108)**
DUCG-EHBSA vs DDAG	**+(0.0254)**
A&O vs DDAG	=(0.3817)
OVO vs OVA	=(0.4953)
OVO vs DUCG-RAND	=(0.8738)
DUCG-RAND vs DDAG	=(0.9433)
DUCG-EHBSA vs OVA	=(0.9433)

better behaviour and in almost all the experiments they get significant improvements comparing with the other methods. We also want to emphasize the results achieved by DUCG-RAND, since several times shows better behaviour than OVA, DDAG and A&O.

4.6.2. Computational Load

In order to complete the experimental study we have performed another comparison analysing the computational cost of each method. To do so, we have calculated the testing time (Table 15) and the number of classifiers (Table 16) that each method needs using Multilayer Perceptron as base classifier. Among all the base classifiers, Multilayer Perceptron has been the selected one because it obtains the best accuracy and kappa results. Nevertheless, the obtained conclusion can be extended to the remaining base classifiers.

The results obtained in Tables 15 and 16 show that OVA, DDAG and A&O are the fastest methods and which need less classifiers. DUCG-RAND needs slightly more time and classifiers. On the other hand, OVO requires the most testing time and uses the most classifiers. Finally, in the case of DUCG-EHBSA, although it needs few number of classifiers, the testing time that it spends is between OVO and the rest methods.

4.7. Discussion

Regarding the obtained results, we emphasize the following:

- OVO and the proposed method DUCG-EHBSA obtain the best results. In the majority of the cases the best result of each database is obtained by one of these methods and they always achieve the best rank and mean. Their improvement is more clear for the databases with more classes. Moreover, the statistical analysis reinforces these conclusions.
- Considering only the accuracy, OVO performs quite well with all base classifiers, however, when kappa is considered, DUCG-EHBSA offers better results. After we have analysed the obtained results in several databases,

Table 15: Comparison of the testing time of 6 methods (in milliseconds)

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	10.1	14.8	8.7	11.1	10.1	12.8
Vehicle	54.3	54.2	49.2	50.7	53.1	51.8
Annealing	71.1	122.7	59.6	82.9	77.1	80
Gesture	6301	6548.1	6262.3	6360.5	6363.3	6363.8
Nursery	1350.1	1445.8	1342.7	1379.6	1376.3	1375.4
PageBlocks	895.5	918.6	900.6	906.6	908.2	892.9
Autouniv	10803.1	13010.3	10685.4	11063.6	11084.3	11373.2
Dermatology	22.5	42.1	20.6	31	25.3	25.9
Flare	20.4	37.6	20	22.5	21.7	23.3
Glass	2.3	4.7	3.3	2.7	2.7	3.1
WineRed	90.8	107.7	88.7	91.5	92.1	93.5
Satimage	4517.8	4878.8	4511.4	4573.4	4584.1	4630
ImageSeg	315.6	390.1	313.7	323.1	328.6	325
Shuttle	52334.5	53063.4	51657.7	52150.8	51913.3	51908
WineWhite	812.6	892.5	812.4	820.7	826.9	823
Zoo	1.6	4	1.4	1.9	2	2
Ecoli	6	10.3	6.6	8.8	6	6.3
Optdig	4030.5	7241	3976.9	4135.8	4366	4495.9
Pendig	2995	3764.3	2994.2	3027.5	3099.9	3128.9
Yeast	48.2	101.2	47.6	49.1	54.3	60.3
Pokerhand	13555.7	14727.4	13534.9	13649.3	13642.7	13922.9
Vowel	48.6	116	49.4	49.5	59.2	62.7
Arrhythmia	1916.5	10039.3	1763.3	2046	2541.8	2835.1
Chess	2552	16383.8	2723.9	2662.6	3736.1	13103.4
Soybean	366.6	3720.8	399.8	387.7	544.2	714.1
Letters	11047.8	30520.8	11399.1	10950	12810.5	25233.9
Abalone	520.2	3067.5	571.4	530.9	773.4	5527.4
Mean	4247.8	6341.8	**4229.8**	4273.0	4418.6	5447.2

Table 16: Number of classifiers used by different methods

Database	OVA	OVO	DDAG	A&O	DUCG-RAND	DUCG-EHBSA
Car	4	6	3	5	4.15	4.84
Vehicle	4	6	3	5	4.28	4.52
Annealing	5	10	4	6	5.66	5.64
Gesture	5	10	4	6	5.71	5.84
Nursery	5	10	4	6	5.45	5.68
PageBlocks	5	10	4	6	5.54	5.93
Autouniv	6	15	5	7	6.99	7.22
Dermatology	6	15	5	7	7.26	7.42
Flare	6	15	5	7	6.96	7.77
Glass	6	15	5	7	7.04	8.32
WineRed	6	15	5	7	6.99	7.97
Landsat	6	15	5	7	6.98	8.03
ImageSeg	7	21	6	8	8.65	9.13
Shuttle	7	21	6	8	8.57	8.86
WineWhite	7	21	6	8	9.03	9.89
Zoo	7	21	6	8	8.77	8.88
Ecoli	8	28	7	9	9.56	10.31
Optdig	10	45	9	11	13.27	13.92
Pendig	10	45	9	11	13.16	14.01
Yeast	10	45	9	11	13.13	14.58
Pokerhand	10	45	9	11	13.11	15.60
Vowel	11	55	10	12	14.81	15.81
Arrhythmia	13	78	12	14	17.21	18.88
Chess	18	153	17	19	25.06	28.70
Soybean	19	171	18	20	26.18	27.65
Letters	26	325	25	27	36.80	38.31
Abalone	28	378	27	29	36.11	41.56
Mean	9.44	59.04	**8.44**	10.44	12.09	13.16

we conclude that selecting the best pairwise comparison is beneficial for the unbalanced problems. One indicative of this behaviour is that in several unbalanced databases OVO obtains better accuracy than DUCG-EHBSA, while it obtains worst result in kappa. These databases are indicated with ”*” in the Tables 4, 6, 8 and 10. The reason of this fact is that trying to select the best class order, the minority classes are more likely to be compared with those classes that are easier to distinguish.

- Depending on the base classifier the results vary. Although the majority of the papers in the literature use an unique base classifier (usually SVM) we use other extra base classifiers in order to obtain a better view of the proposed approach. In fact, the results show that when SVM is used, our approach, DUCG-EHBSA, shows the best performance. On the other hand, for Multilayer Perceptron DUCG-EHBSA and OVO are the most remarkable strategies. Finally, for Ripper and C4.5 base classifiers, the statistical tests also conclude that DUCG-EHBSA and OVO are the most robust ones, however, it is worth mentioning that OVO obtains the best mean and the best result in the majority of the databases with these base classifiers.
- Although OVO and DUCG-EHBSA are the strategies that need more classification time, as the results are considerably in favour of them, we consider that their good performance compensates their computational cost. However, if Occam razor's principle (in equal conditions simplest model is selected) is used for tie-breaking, DUCG-EHBSA would be selected since it needs less testing time and classifiers than OVO.
- The achieved results show the importance to sort the classes in the proper order since DUCG-EHBSA shows better performance than DUCG-RAND. DUCG-EHBSA obtains better results in most of the databases and it outperforms DUCG-RAND in the mean and rank of all the experiments.
- DUCG-RAND obtains interesting results since it obtains better rank and mean than OVA, DDAG and A&O in almost all the experiments, besides several statistical tests show its better performance. Moreover, with SVM it obtains better kappa mean and rank than OVO and in Multilayer Perceptron it obtains the same kappa rank.
- The state-of-the-art approaches that try to reduce the number of classifiers in OVO obtain poor results. We refer to DDAG and A&O. Not considering OVA, they obtain the worst mean and rank result in almost all the experiments. Moreover, they are significantly improved several times by other methods. Although they obtain competitive results with the databases with less classes, they show a worst tendency in the databases with high number of classes where they obtain considerable worse mean than OVO and DUCG-EHBSA.

5. Conclusion

In this work, we have presented a new method called Decision Undirected Cyclic Graph that reduces the number of classifiers in OVO. We have carried out our experiments for four different Machine Learning algorithms and we have compared the obtained results with those obtained with several state-of-the-art methods. We have carried out this experiments using two different metrics to calculate the accuracy.

We conclude that DUCG-EHBS is a promising decomposition strategy, since the experimental results show that OVO and DUCG-EHBSA are the most robust methods. We show that the best aggregation within a problem depends on the base classifier that is considered, since SVM works better when DUCG-EHBSA decomposition is used, and in Ripper, C4.5 and Multilayer Perceptron both decomposition strategies are equivalent. Moreover, we also have shown that DUCG-EHBSA obtains better performance than OVO for kappa metric.

We have obtained several interesting conclusions, one of the most important one is the good behaviour of DUCG-EHBSA in problems with large amount of classes, where it performs as well as OVO, whereas other state-of-the-art strategies that attempt to reduce the number of classifiers show their weakness. Moreover, the new proposal needs less testing time and less classifiers to take the final decision than OVO.

As we present a novel strategy to reduce the number of classifiers it gives the possibility for future works. One option is to try to reduce the classification time using other faster strategies, such as genetic algorithms or Kruskal graph constructor algorithm, to obtain the class order. Other option is to calculate the class order fitness using different strategies, for example class separability measures. And another option is to observe the performance of this strategy incrementing the number of edges in each node.

Acknowledgements

The authors gratefully acknowledge J. Ceberio for his assistance during the work. The work described in this paper was partially conducted within the Basque Government Research Team Grant IT313-10. I. Mendialdua holds a Grant from Basque Government.

References

[1] Anand, R., Mehrotra, K., Mohan, C.K., Ranka, S., 1995. Efficient classification for multiclass problems using modular neural networks. Neural Networks, IEEE Transactions on 6, 117–124.

[2] Bagheri, M.A., Gao, Q., Escalera, S., 2012. Efficient pairwise classification using local cross off strategy, in: Advances in Artificial Intelligence. Springer, pp. 25–36.

[3] Bautista, M.Á., Escalera, S., Baró, X., Radeva, P., Vitriá, J., Pujol, O., 2012. Minimal design of error-correcting output codes. Pattern Recognition Letters 33, 693 – 702.

[4] Ben-David, A., 2007. A lot of randomness is hiding in accuracy. Engineering Applications of Artificial Intelligence 20, 875–885.

[5] Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A., 2012. A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. Progress in Artificial Intelligence 1, 103–117.

[6] Cohen, J., 1960. A coefficient of agreement for nominal scales. Educational and Psychological Measurement 20, 37–46.

[7] Cohen, W.W., 1995. Fast effective rule induction, in: In Proceedings of the Twelfth International Conference on Machine Learning, Morgan Kaufmann. pp. 115–123.

[8] Dietterich, T.G., Bakiri, G., 1995. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2.

[9] Fei, B., Liu, J., 2006. Binary tree of svm: a new fast multiclass training and classification algorithm. Neural Networks, IEEE Transactions on 17, 696–704.

[10] Frank, A., Asuncion, A., 2011. Uci machine learning repository, 2010. URL http://archive. ics. uci. edu/ml .

[11] Friedman, J., 1996. Another approach to polychotomous classifcation. Technical Report. Technical report, Stanford University, Department of Statistics.

[12] Fürnkranz, J., 2002. Round robin classification. The Journal of Machine Learning Research 2, 721–747.

[13] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., Herrera, F., 2011. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. Pattern Recognition 44, 1761–1776.

[14] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., Herrera, F., 2013. Dynamic classifier selection for one-vs-one strategy: Avoiding non-competent classifiers. Pattern Recognition 46, 3412 – 3424.

[15] García, S., Fernández, A., Luengo, J., Herrera, F., 2010. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences 180, 2044–2064.

[16] Garcia-Pedrajas, N., Ortiz-Boyer, D., 2006. Improving multiclass pattern recognition by the combination of two strategies. Pattern Analysis and Machine Intelligence, IEEE Transactions on 28, 1001–1006.

[17] García-Pedrajas, N., Ortiz-Boyer, D., 2011. An empirical study of binary classifier fusion methods for multiclass classification. Information Fusion 12, 111–130.

[18] Ghaffari, H.R., Yazdi, H.S., 2013. Multiclass classifier based on boundary complexity. Neural Computing and Applications , 1–9.

[19] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The weka data mining software: an update. ACM SIGKDD Explorations Newsletter 11, 10–18.

[20] Hastie, T., Tibshirani, R., 1998. Classification by pairwise coupling. The annals of statistics 26, 451–471.

[21] Hong, J.H., Min, J.K., Cho, U.K., Cho, S.B., 2008. Fingerprint classification using one-vs-all support vector machines dynamically ordered with naı¨ ve bayes classifiers. Pattern Recognition 41, 662–671.

[22] Hsu, C.W., Lin, C.J., 2002. A comparison of methods for multiclass support vector machines. Neural Networks, IEEE Transactions on 13, 415–425.

[23] Hüllermeier, E., Vanderlooy, S., 2010. Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting. Pattern Recognition 43, 128–142.

[24] Kijsirikul, B., Ussivakul, N., 2002. Multiclass support vector machines using adaptive directed acyclic graph, in: Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on, IEEE. pp. 980–985.

[25] Ko, J., Byun, H., 2003. Binary classifier fusion based on the basic decomposition methods, in: Proceedings of the 4th international conference on Multiple classifier systems, Springer. pp. 146–155.

[26] Kumar, M.A., Gopal, M., 2011. Reduced one-against-all method for multiclass {SVM} classification. Expert Systems with Applications 38, 14238 – 14248.

[27] Kumar, S., Ghosh, J., Crawford, M.M., 2002. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Pattern Analysis & Applications 5, 210–220.

[28] Lorena, A.C., de Carvalho, A.C., 2010. Building binary-tree-based multiclass classifiers using separability measures. Neurocomputing 73, 2837 – 2845.

[29] Platt, J.C., 1999. Fast training of support vector machines using sequential minimal optimization, in: Schölkopf, B., Burges, C.J.C., Smola, A.J. (Eds.), Advances in kernel methods. MIT Press, pp. 185–208.

[30] Platt, J.C., Cristianini, N., Shawe-Taylor, J., 2000. Large margin dags for multiclass classification. Advances in neural information processing systems 12, 547–553.

[31] Pujol, O., Radeva, P., Vitria, J., 2006. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. IEEE Trans. Pattern Anal. Mach. Intell. 28, 1007–1012.

[32] Quinlan, J.R., 1993. C4. 5: programs for machine learning. volume 1. Morgan kaufmann.

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

[34] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1985. Learning internal representations by error propagation. Technical Report. DTIC Document.

[35] Tsutsui, S., 2002. Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram, in: Parallel Problem Solving from Nature—PPSN VII. Springer, pp. 224–233.

[36] Wu, T.F., Lin, C.J., Weng, R.C., 2004. Probability estimates for multi-class classification by pairwise coupling. The Journal of Machine Learning Research 5, 975–1005.