Konputazio Zientziak eta Adimen Artifizialaren Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Contributions on Distance-Based algorithms, Multi-Classifier Construction and Pairwise Classification

by

Iñigo Mendialdua Beitia

Supervised by Basilio Sierra

Dissertation submitted to the Department of Computer Science and Artificial Intelligence of the University of the Basque Country (UPV/EHU) as partial fulfilment of the requirements for the PhD degree in Computer Science

Donostia - San Sebastián, April 2015

To my family
and friends

Laburpena

Aurkezten den ikerketa lan honetan saikapen atazak landu dira, non helburua, sailkapen gainbegiratuaren artearen-egoera aberastea izan den. Sailkapen gainbegiratuaren zenbait estrategi analizatu dira, beraien ezaugarri eta ahuleziak aztertuz. Beraz, ezaugarri positiboak mantenduz, ahuleziak hobetzeko saiakera egin da. Hau burutu ahal izateko, sailkapen gainbegiratuaren zenbait estrategi konbinatzeaz gain, zenbait bilaketa heuristiko ere erabili dira.

Sailkapen gainbegiratuko 3 ikerketa lerro desberdinetan burutu dira ekarpenak. Aurkezten diren lehenengo proposamenak, K-NN algoritmoan zentratzen dira, honen zenbait bertsio aurkezten direlarik. Ondoren sailkatzaileen konbinaketarekin erlazionatutako beste lan bat aurkezten da. Eta azkenik, binakako sailkapenaren zenbait estrategi berritzaile proposatzen dira. Ekarpen hauek aldizkari edo konferentzi internazionaletan publikatuak edo bidaliak izan dira.

Buruturiko experimentuetan, proposatutako algoritmoak artearen-estatuan aurkituriko zenbait algoritmorekin konparatu dira, emaitza interesgarriak lortuaz. Honetaz gain, emaitza hauetatik ondorio esanguratsuak eskuratzeko asmoz, test estatistikoen erabilera ere burutu da.

Summary

The work presented here deals with classification problems, where the aim is to incorporate new Supervised Classification methodologies to the state-of-the-art. Several Supervised Classification methods are studied, analyzing their positive points and weaknesses. In order to do so, in addition to combining different methods of Supervised Classification, Evolutionary Algorithms have also been used.

Contributions of Supervised Classification are presented in 3 research lines. The first contributions are focused on K-NN algorithm, proposing several versions. Next, another work related to the classifier combination is presented. Finally, several innovative Class Binarization strategies are proposed. All these contributions have been published or submitted to international journals or conferences.

In the performed experiments, the proposals are compared with several state-of-the-art algorithms, obtaining interesting results. Moreover, statistical tests are applied in order to obtain meaningful conclusions from the obtained results.

Acknowledgments

Lehenik eta behin, nere zuzendaria izan den Basilio Sierrari eman nahiko nizkioke eskerrak. Bere gidaritza gabe lan hau burutzea ezinezkoa litzateke. Zure pazientzia, optimismo eta animoak, gomendio teknikoetaz gain oinarrizkoak izan dira. Mila esker benetan.

Também eu quero agradecer, com o meu pobre Português, para todos os LIAAD-dianos por me permitir completar uma estadia maravilhosa como um visitante em seu grupo de pesquisa da Universidade do Porto.

Eskertza bereziak urte guzti hauetan bulegoa partekartu dudan Robotika eta Sistema Autonomoen Ikerketa Taldeko partaide guztiei: Elena, Ekaitz, Ali, Aitzol, Igor, Enara, Otzeta, David... y también a los miembros de otros grupos de investigación de la Facultad de Informática. El buenísimo ambiente de trabajo, la continua disposición a ayudar y el apoyo recibido han sido fundamentales.

Josu Ceberiori eta Ander Iruetaguenari ere eskerrak eman nahiko nizkieke: Josuri bilaketa heuristikoetan eskeinitako laguntzagatik eta Anderri eskeinitako laguntza guztiagatik.

I am grateful to the Basque Government for the financial support during four years. In addition, this work has been partially conducted within the Basque Government Research Team Grant IT313-10 and the University of the Basque Country UPV/EHU.

Nere kuadrilla eta lagunak ere eskertu nahiko nituzke, momentu zailetan emandako laguntza ezinbestekoa izan baita. Bereizki Gari, Iban eta Enekongandik jasotako babesa oso garrantzitsua izan da.

Ezin naiteke Itziar Gonzalez-Diosetaz ahaztu, bere laguntza, animo eta pazientziagatik azken bi urte hauetan.

Eta azkenik, nere eskertza handiena azken urte hauetan nere humore txarra gehien sufritu dutenei: Aita, Ama eta Andoniri. Tesi hau beraiena da ere.

Mila esker guztiei!
Muchas gracias a todos!
Muito obrigado a todos!

Contents

Part I

Framework of the Project

1 Introduction

Although we are unaware in our daily life, we apply several complex activities to routines that we carry out intuitively: when we are selecting the proper key to open a door, when we are driving and we recognize the traffic sign from a different angle or distances or when we select the shortest path. To do so, we obtain information from the environment, our brain processes the information and decides how to react in each situation.

This human behaviour has been extended to computers, in what it is known as Artificial Intelligence. The aim of Artificial Intelligence is to learn how to solve complex problems that require certain skills beyond mere calculation capacity. There exist several lines of research within Artificial Intelligence where the classification task is one of the most popular research lines.

Figure 1.1, presents a diagram of how a classification task is organized. When we have to classify something we usually base our decisions on our experiences. However, as computers do not have experiences, in those problems the experiences are represented by a set of examples or instances. Based on these examples, the learning process is developed and a classifier which is going to be used to classify future unseen instances is created. In this thesis project we will focus on this step, which is commonly known as Machine Learning.

Machine Learning problems are mainly divided into two categories: Unsupervised Classification and Supervised Classification.

Unsupervised Classification, also known as clustering, is focused on finding the classes of the problem. In Unsupervised Classification a set of instances is provided where the class label of those instances is unknown. The aim of Unsupervised Classification problems is to discover groups of similar examples within the data and to assign the same class label to the instances of each group.

Supervised Classification focuses on the problem in a different way. In Supervised Classification a set of well labelled instances is provided, this set is also known as training data. Based on the those instances, the Supervised Classification strategies build a classifier, which is used to classify new unla-

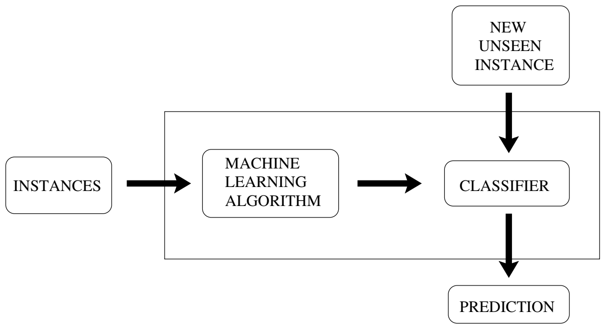


Fig. 1.1: Classification task

belled instances. The error obtained with the training data is a guide to build the model. However, in order to check the error rate, it is better to use a set of instances that are not used to train the model.

The aim is to construct a classifier that will be able to classify every instance as belonging to its real class. In those problems where the classes are clearly different it is possible to obtain this result. However, this does not happen in many real problems. Because of that, there exist many types of classifiers that are based on different paradigms. Moreover, there are also a lot of proposals that try to improve the performance of those classifiers. Some of them introduce new variations of existing strategies, other approaches choose to combine some of them, and others decide to divide the original problem into several sub-problems that are easier to solve.

The main objective of this thesis project, is to perform a study of different algorithms within the Supervised Classification area, trying to incorporate new ideas that contribute to the state-of-the-art. Methodological contributions have been presented in the following research lines:

Distance-Based Algorithms: a study of the different variations of these algorithms has been performed and several versions are presented that can improve standard algorithm performance in some kind of classification problems.

Multi-Classifiers: in this work a new variation of the Stacked Generalization strategy is presented.

Class Binarization: several versions are presented that try to solve some of the weaknesses found in the literature. In order to do so, it has been proposed to combine several Artificial Intelligence strategies.

1.1 Overview of the dissertation

The document is divided into four parts:

- The first part consists of 8 chapters and describes the framework that has been developed in the research work. It starts with a description (Chapter 2) of the most relevant concepts and procedures of Supervised Classification and the classifiers that have been used in the experiments of the dissertation. Chaper 3 describes the Distance-Based algorithms, focusing on K-NN method, one of the most simple methods but which provides satisfactory results. Chapter 4 introduces the Multi-Classifiers, where the most popular strategies are explained. In Chapter 5 the most relevant Class Binarization strategies are presented focusing especially on Pairwise Classification or One versus One strategy. In Chapter 6 a brief description of the contributions of the dissertation are shown. Finally Chapter 7 exposes the final conclusion of the work and further work.
- The second part presents those publications related to Distance-Based Algorithms.
- The third part presents those publications related to Multi-Classifiers.
- The fourth part presents those publications related to Class Binarization.

2

Supervised Classification

2.1 Introduction

Supervised Classification is one of the most important tasks in the field of Pattern Recognition, and it is used to solve problems of almost every type of domains. Supervised Classification is based on the idea of obtaining knowledge based on past experience. To do so, in Supervised Classification, there is a set of well-labelled instances, called training data, that describes the experiences. Those instances contain the available collection of relevant characteristics extracted from an object and a class that labels an object.

Let training data $TR = \{x_i, \theta_i\}_{i=1}^{N}$ denote a set of N well-labelled instances, where x_i represents the i-th individual feature vector and θ_i represents the class the individual belongs to. In the particular case of the K-class problem, being $\theta \in \{1, .., K\}$, the class label is commonly defined as an integer. Table 2.1 illustrates an example of training data.

Instance	X_1	X_2	$\ldots$	X_L	$Class$
1	x_{11}	x_{21}	$\ldots$	x_{L1}	θ_2
2	x_{12}	x_{22}	$\ldots$	x_{L2}	θ_K
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
N	x_{1N}	x_{2N}	$\ldots$	x_{LN}	θ_1

Table 2.1: Example of Training Data

Based on the TR, the Supervised Classification techniques create a "general rule" that is also known as a classifier. Then, when a new unknown instance to be classified arrives, the classifier assigns one of the different previously defined K classes to the unseen instance.

Although the main objective is to create a classifier that classifies correctly as many instances as possible, there are other criteria that should be considered in order to evaluate a classifier. In the following paragraphs we will describe some of the most important ones:

- Accuracy: this measures the percentage of testing set examples correctly classified by the classifier.
- Speed: the computational cost involved in generating the classifier or in predicting new unseen data.
- Interpretability: this is closely related to the concept of explanation; an interpretable classifier ought to be able to explain its predictions.
- Simplicity: it is preferable to use the simplest classifier among the classifiers with the same accuracy.

2.2 Types of Classifiers

The aim of the Supervised Classification techniques is to build a model which will return a prediction for the future test instances.

There exist different paradigms to build the model, in this section we explain those which have been used in this project.

2.2.1 Distance-Based Classifiers

The Distance-Based classifiers are based on the idea the similar instances tend to have similar solutions. Because of their simplicity, those classifiers are frequently used and very well-known in the Machine Learning community.

2.2.1.1 K-Nearest Neighbor (K-NN)

The K-Nearest Neighbor (K-NN) is proposed by Fix and Hodges [36]. When an instance to be classified arrives, the K-NN algorithm selects its K nearest instances from the training data. The most represented class among those K instances is assigned to the new instance.

2.2.2 Decision Trees

Decision Trees are one of the most popular classifiers due to several reasons: the simplicity of the model, explanation of the decision, the possibility to represent it graphically and the decision speed. The philosophy of Decision Trees is to divide the classification space into several areas and a class is assigned to the patterns that belong to each area.

Usually Decision Trees are represented as a tree structure (see Figure 2.1) that is formed by nodes. To classify an instance, it is started in the root-node. Depending on the value of the predictor variable which is asked for, the instance is moved down the tree to the different internal-nodes until it arrives at a leaf-node in which there is a class value that is the one assigned to the new instance.

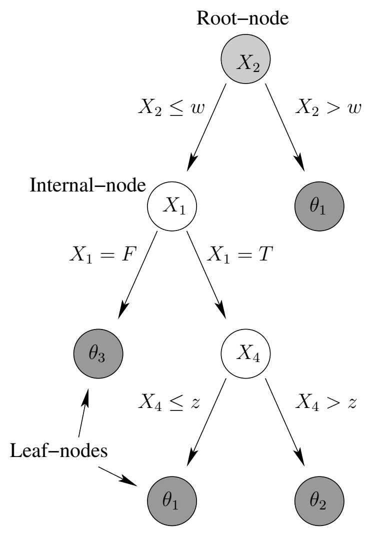


Fig. 2.1: Example of a Decision Tree

2.2.2.1 ID3

One of the first Decision Trees is presented by Quinlan [91] and it is called Itemized Dichotomizer 3 (ID3). ID3 employs a top-down, greedy search through the space of possible branches with no backtracking. Moreover, it uses information gain to decide which attribute goes into a decision node.

ID3 does not prune the expanded tree and it has several weaknesses: it is not able to work with continuous variables, it does not contemplate missing values and does not allow noise in the data.

2.2.2.2 C4.5

C4.5 is an extension of ID3, it is also introduced by Quinlan [92] and it tries to outperform some of ID3's limitations. To achieve this, it performs a post-pruning phase, based on an error based pruning algorithm, and it uses an extension to information gain known as gain ratio.

2.2.3 Rule-Based Classifiers

Rule-Based algorithms generate a set of rules which try to explain the feature space. Each rule is formed by the antecedent ("IF-part") and a consequent ("THEN-part"); the antecedent contains a set of conditions over the different attributes and the consequent contains the class that the rule predicts if the conditions of the antecedent are fulfilled. An example of the structure of the rules is illustrated in Figure 2.2.

$$\begin{array}{lll} Rule_1 & IF(X_2 > 3 \,\&\, X_3 < 0) & THEN\ \theta_1 \\ Rule_2 & IF(X_1 < 1 \,\&\, X_2 > 2) & THEN\ \theta_3 \\ \ldots & & \\ Rule_M & IF(X_4 == True) & THEN\ \theta_K \end{array}$$

Fig. 2.2: Example of rules structure

The rules can receive a specific order, such as, it is assigned the class of the first rule that its antecedent is fulfilled.

There are different algorithms for rule generation:

2.2.3.1 1R

1R algorithm is proposed by Holte [56] and as its names indicates, 1R is a program that learns 1-rule from the examples. It creates a rule for each attribute and then it selects the rule with the minimum error.

2.2.3.2 CN2

The CN2 algorithm is introduced by Clark and Niblett [14] and it constructs rules so that they are evaluated in a specific order. CN2 consists of two main procedures: the search algorithm performing a beam search in order to find a good rule, and the control algorithm for repeatedly executing the search.

2.2.3.3 Ripper

RIPPER is proposed by Cohen [17] as an extension of IREP and is the acronym for Repeated Incremental Pruning to Produce Error Reduction.

IREP orders the classes by increasing frequency. It finds the rule that separates the minority class from the remaining classes. Those instances covered by the learnt rule are removed from the dataset. In the next iteration the second minority class is separated from the remaining ones. This process is repeated until a single class remains.

RIPPER includes a new pruning and stopping criteria as well as post-processing phase in order to optimize the set of learned rules.

2.2.4 Bayesian Classifiers

Under this denomination, the classifiers that use the Bayesian reasoning are grouped in order to assign to a new instance the most likely class value.

2.2.4.1 Bayesian Networks

Bayesian Networks are a graphic representation of the conditional dependencies between a set of random variables. A Bayesian Network represents a pair (G,P). G is a directed acyclic graph where each node represents a variable and $P = P(x_1|\pi_1, \ldots, x_n|\pi_n)$ is a set of n conditional probability functions, one for each variable, and π_i is a set of parents of the node x_i in G. P defines a probability function associated with the following factorization:

$$P(x) = \prod_{i=1}^{n} P(x_i|\pi_i) \tag{2.1}$$

Bayesian Network models stand out because of their interpretability: the probability relationships allow to understand the influences and dependencies among variables.

2.2.4.2 Naive Bayes

Naive Bayes is introduced by Cestnik [84]. Naive Bayes classifier assumes that the predictive variables are independent among them given the class variable. Hence, the posterior probability can be obtained by the product of the individual conditional probabilities of each attribute given the class node.

$$P(x\epsilon\theta_j) = P(\theta_j) \prod_{i=1}^{n} P(x_i|\theta_j) \tag{2.2}$$

2.2.4.3 Naive Bayes Tree

This method is introduced by Kohavi [65] and it is a hybrid algorithm. A Decision Tree is generated but in each leaf a Naive Bayes is constructed with the instances of the node.

2.2.5 Neural Networks

Neural Networks are a parallel distributed processing structure inspired by human brain performance.

A Neural Network consists of a set of interconnected nodes (neurons). Those connections receive a weight, where the higher the value the stronger the connection between the nodes is. Each neuron receives inputs from other neurons and the activation of a neuron is determined by a mathematical function, for instance, a weighted sum of the inputs, that determines the output of the neuron.

One of the most common Neural Networks is the Perceptron. A single Perceptron has two layers: the input layer and the output layer. One of the main disadvantages of the single Perceptron is that it is not able to solve non-linearly separable problems

2.2.5.1 Multilayer Perceptron

The Multilayer Perceptron [99] is a feedforward artificial neural network which is trained with a back-propagation learning algorithm.

The Multilayer Perceptron is able to deal with non-linearly separable problems introducing a new layer (see Figure 2.3). While the single Perceptron only has two types of layers (input and output), the Multilayer Perceptron introduces a third type, the hidden layer. When an instance to be classified is presented to a Multilayer Perceptron, the input nodes take the information, this information is then passed throughout the hidden nodes until it reaches the output nodes. The main disadvantage of the Multilayer Perceptron is the interpretability of the algorithm, it does not contain an easily understood representation of the knowledge.

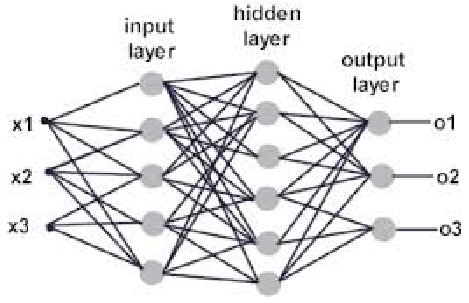


Fig. 2.3: Example of Multilayer Perceptron

2.2.6 Support Vector Machines (SVM)

Support Vector Machines (SVM) are the most common Kernel based method and are proposed by Vapnik [7] . SVM performs classification by finding the hyperplane that maximizes the margin between the two classes.

SVM can be seen as an extension of Maximal Margin Classifiers, classifiers proposed to solve linear decision boundaries. The motivation behind the SVM idea is to allow non-linear decision boundaries. To do so, the Support Vector Machines enlarges the feature space through the use of functions called kernels.

The kernel function can be seen as a similarity function between two instances. There are different kinds of kernel functions but the most popular are the polynomial and the Gaussian kernels.

In a problem of two classes (θ_0, θ_1), when an instance x' to be classified arrives, the kernel function, $k(a, b)$, that measures the similarities is applied between the new instance and all the training data instances $\{x_1, ..., x_n\}$. Next the formula $w_0 + w_1 * k(x', x_1) + w_2 * k(x', x_2)...$ is applied, where the w_i values are fixed in the training phase of the algorithm. If the result is higher than 0, θ_0 is predicted, otherwise θ_1 is the predicted class value.

2.3 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 [6] shows that several hits could be attributed to chance, and in order to compensate these random hits, he proposes to use Cohen's Kappa metric [16].

- Classification rate: this is also called accuracy. Among all the classified instances, it calculates the proportion of well classified instances.
- Cohen's Kappa [16]: this metric tries to calculate the portion of hits that can be attributed to the classifier itself and are not obtained by chance.

  $$kappa = \frac{P_0 - P_c}{1 - P_c} \tag{2.3}$$

  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 2.3 is equivalent to this one:

  $$kappa = \frac{n \sum_{i=1}^{K} h_{ii} - \sum_{i=1}^{K} T_{ri} T_{ci}}{n^2 - \sum_{i=1}^{K} T_{ri} T_{ci}} \tag{2.4}$$

  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 perform at least as well as random, so in general they score Kappa higher than 0.

2.4 Classification error estimators

In a classification problem when the dataset of well classified instances is available but there are no independent testing samples available, there are several strategies, known as Classification Error Estimators, that allow to calculate an estimation error given a classifier. In this sub-section, some of the most popular Classification Error Estimators used in Machine Learning are presented:

2.4.1 Bootstrap and 0.632 bootstrap

The bootstrap estimator [30] is based on the statistical procedure of sampling with replacement. The sampled instances are used as training data and the original data as testing data.

The 0.632 bootstrap estimator [31] consists of the resubstitution estimator [24] with a bias correction based on bootstrap samples. The 0.632 bootstrap is recommended by Rodriguez et al. [98] for problems with low computational complexity.

2.4.2 Hold-out and repeated hold-out

The hold-out method [71] partitions the dataset into two mutually exclusive subsets: one subset is the training data and the other is the testing data. The proportion of the data in each subset can vary, but it is common to hold-out 1/3 of the data for testing and use the remaining 2/3 for training.

Repeated hold-out estimator is based on iterating a random hold-out process several times.

2.4.3 Cross-validation and repeated cross-validation

In k-fold cross-validation [55], the dataset is randomly split into k mutually exclusive subsets. Each of these partitions is known as folds and each fold is approximately of equal size. The classifier is trained and tested k times; each time, $k - 1$ folds are used as training set while the remaining fold is used as testing set. Commonly the k parameter is given a value of 10, which is recommended by Kohavi [66].

As in repeated hold-out, the repeated k-fold cross-validation estimator is based on iterating a random k-fold cross-validation process several times. In this case the 5x2-fold cross-validation is commonly used, which is recommended by Dietterich [25].

2.4.4 Leave-one-out

The leave-one-out estimator is a k-fold cross-validation estimator with k equal to the number of instances in the dataset. In each iteration, each instance is used as testing data, and the learning method is trained on all the remaining instances.

2.5 Statistical Comparisons of Classifiers

It is often necessary to compare several classifiers over the same problem in order to decide which of them achieves better results. However, in order to obtain a meaningful decision, it is necessary to carry out a statistical analysis.

The statistical tests considers that the null hypothesis being tested is that all methods obtain similar results with non-signicant differences. Then the rejection of the null hypothesis means that there exist statistical differences among the classifiers that are compared.

Although there exist a variety of methods, in this sub-section only two non-parametric statistical tests are presented: the Wilcoxon test, which is used to compare two classifiers over multiple databases, and the Friedman test and Iman-Davenport extension, which is used to compare multiple classifier over multiple databases.

2.5.1 Wilcoxon Test

The Wilcoxon signed-rank test [118] is a non-parametric test. The use of this statistical test is suggested by Demšar [21] to compare two algorithms based on multiple data sets. It ranks the accuracy differences (according to their absolute values) of two classifiers for each data set, ignoring the signs.

Let $R+$ be the sum of ranks for the data sets on which the first classifier obtains the best results and $R-$ the sum of the ranks for the opposite. And let N be the number of the data sets. Then, the statistic

$$z = \frac{min(R+, R-) - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{2.5}$$

has approximately a standard normal distribution, $\mathcal{N}(\mu = 0, \sigma = 1)$.

2.5.2 Friedman test and Iman-Davenport extension

Friedman test [39] and Iman-Davenport [61] extension are also non-parametric tests and they are recommended by García et al. [43] to observe if statistical differences exist among several classifiers over several data sets.

The Friedman test ranks the classifiers for each data set separately and compares the average ranks of them:
$R_i = \frac{1}{N}\sum_{j=1}^{N} r_i^j$
being r_i^j the rank of the ith of n algorithms on the jth of N data sets.
Friedman's statistics

$$X_F^2 = \frac{12N}{n(n+1)}[\sum_{i=1}^{n} R_i^2 - \frac{n(n+1)^2}{4}] \tag{2.6}$$

follows, under the null hypothesis, a chi-square distribution with $(n-1)$ degrees of freedom.

Iman and Davenport demonstrate that Friedmans X_F^2 presents a conservative behaviour and propose a better statistic

$$F_F = \frac{(N-1)X_F^2}{N(n-1-)X_F^2} \tag{2.7}$$

which follows a F-distribution with $(n-1)$ and $(n-1)(N-1)$ degrees of freedom.

Under the null-hypothesis, these algorithms state that all the classifiers are equivalent. On the other hand, the rejection of the null-hypothesis implies the existence of statistical differences among the classifiers. In order to find the specific pairwise comparisons which produce differences, a post-hoc test must be carried out. There are different post-hoc procedures. In [44], García and Herrera suggest using the Shaffer post-hoc procedure.

The Shaffer post-hoc procedure [103] is an extension of Holm's procedure and follows a step down method. Let $p_1, ..., p_m$ be the ordered p-values (smallest to largest) and $H_1, ..., H_m$ be the corresponding hypothesis. Shaffer's procedure rejects H_i if $p_i \leqslant \frac{\alpha}{t_i}$, where t_i is the maximum number of hypothesis which can be true given that any $(i-1)$ hypothesis is false.

3 Distance-Based Algorithms

Distance-Based algorithms assume that there is a metric that measures the distance between two instances and they consider the assumption that the closer two instances are, the more similar they are. In short, they measure the similarity among the instances based on their distance. The metric can vary depending on the problem. The most common distance is the Euclidean Distance, although other distances such as Manhattan or Mahalanobis are also used.

One of its main advantages is its conceptual simplicity, since the fact that two instances are similar is instinctive. Because of that, this kind of algorithm is widely used in the literature.

Among the Distance-Based algorithms, the method called K Nearest Neighbor (K-NN) is the most popular.

3.1 K Nearest Neighbors (K-NN)

K-NN is the extension of the Nearest Neighbors (NN) classification method, and it assigns the category of its nearest neighbor to the new case. The first formulation of a rule of the NN type and primary previous contributions to the analysis of its properties is presumed to have been made by Fix and Hodges [36].

Let $\{x_i, \theta_i\}_{i=1}^{N}$ denote a training set of N well-labelled examples, where x_i represents i-th individual feature vector, and θ_i is the class which the individual belongs to.

Consider a new case (x, θ), where θ is unknown. To estimate θ, the information contained in the set of correctly classified examples is used. In this case, the NN method assigns the class of the most similar point to this new point. To do so, a distance d is needed, which calculates the dissimilarity between the cases. So, x' is called the Nearest Neighbor (NN) of x if

$$\min_{i=1,\dots,n} d(x_i, x) = d(x', x)\,. \tag{3.1}$$

An immediate extension to this decision rule is the so called K-NN approach [18], which assigns to the candidate x the class which is most frequently represented among its K nearest neighbors.

The K number is an important parameter. If it is too small, the result is sensitive to outliers and K-NN will ignore some close points which would be relevant, while if it is too high, further instances can be included in the decision, which frustrates the basic philosophy behind K-NN.

Fig. 3.1 illustrates a 3-NN decision rule. The 3 nearest neighbors of the candidate case are selected, the instances inside the circle. In this case, since two of the three neighbors belong to θ_o, applying the majority vote θ_o is assigned to the new case.

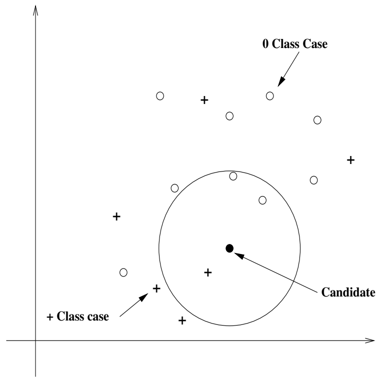


Fig. 3.1: Three Nearest Neighbor Decision Rule

Much research has been devoted to the K-NN rule [19, 117]. One of the most important results is that K-NN has very good asymptotic performance. Broadly speaking, for a very large design set, the expected probability of incorrect classifications (error) R achievable with K-NN is bounded as follows:

$$R^* < R < 2R^* \tag{3.2}$$

where R^* is the optimal (minimal) error rate for the underlying distributions. This performance, however, is demonstrated for the training set size tending to infinity, and thus, it is not really applicable to real world problems in which we usually have a training set of hundreds or thousands of cases, too few for the number of probability estimations to be performed.

3.1.1 K-NN versions

There also exist several versions of the K-NN algorithm. The Instance-Based learning algorithms (IBL) [1] are some of the most popular. The main difference between IBL and K-NN is that IBL is incremental and it treats the noise problem in a specific way.

Other versions propose different ways to obtain the K neighbors. Cleary and Trigg [15] propose to use the entropy-based distance measure, whereas Hastie and Tibshirani [52] suggest computing neighborhoods in directions orthogonal to the local decision boundaries; to do so they use a local linear discriminant analysis.

The dynamic selection of the best K value has also been treated in the literature. Wang et al. [114] introduce a new method that dynamically adjust the number of nearest neighbors based on the statistical confidence, whereas Xie et al. [122] propose to select the K parameter that obtains the highest estimated accuracy applying Naive Bayes classifier.

3.1.2 Weights in K-NN

In order to improve K-NN accuracy, many proposals consider introducing weights in K-NN when classifying. On the one hand, it is proposed to select or to weight the features [50, 100, 111]. On the other hand, the use of probabilistic voting approaches is advocated [83, 107]; the main idea here is to calculate the probability of each case belonging to its own class. A weight is assigned to each case depending on this probability. Another option is to give different weights to each neighbor.

Aiming to deal with the unbalanced dataset problems, Tan [112] develops a new method that assigns different weights to the neighbors. This method assigns higher weight to those neighbors that belong to a class with few solutions. Inversely, a lower weight is assigned to the neighbors that belong to a class with many solutions.

Arguing that the closer instances should be weighted more heavily than the farther ones, Dudani [29] presents a new method which weights each neighbor depending on the distance from the unclassified observation. This method is called the Distance Weighted K Nearest Neighbor (DW-K-NN). Being Dw_j the weight assigned to the j-th neighbor, the following equation describes how it is obtained:

$$Dw_j = \begin{cases} \frac{d_K - d_j}{d_K - d_1}, & for\ d_K \neq d_1 \\ 1, & for\ d_K = d_1 \end{cases} \tag{3.3}$$

where d_K, d_j and d_1 are the distances between the new case and the K-th neighbor, the j-th neighbor and the nearest neighbor, respectively. Note that Dw_j takes values between 0 and 1.

In this work Dudani also proposes a second equation to weight the neighbors based on distance.

$$Dw_j = \frac{1}{d_j + c} \tag{3.4}$$

where the constant c takes a value near to 0 and is added to avoid the situation where the denominator takes 0 value, The drawback of this option is that Dw_j takes very large values for distances to d_j close to zero, and thus in many cases the classification algorithm is reduced to a simple nearest-neighbor rule.

3.1.3 Weakening weaknesses

K-NN also has some drawbacks and several works try to reduce them. For example Hellman [54] criticizes that K-NN algorithm sometimes makes a decision without enough certainty about the answer it is giving, mainly due to weak majority results. In order to avoid that, he presents the (K,L)-Nearest Neighbor rule, where patterns with higher risk of being misclassified are rejected. Other approaches [23, 78] propose to extend K-NN in the belief functions framework to better model the uncertain information.

The low tolerance with outliers is another disadvantage of the K-NN method due to the fact that it considers all data as relevant. In this sense, Shin et al. [105] propose to remove the outliers from the training set. Using this idea, they improve the classification rate for text categorization.

Another weakness of the K-NN method is that in the management of large data sets the computational demands can be high, as it is necessary to calculate a new case's distance with all the training instances. Some approaches try to accelerate the execution [74, 115] to obtain a faster classification. Others, propose a data reduction [34]. A hierarchically structured approach has also been introduced to deal with this problem [97].

3.2 Contributions on Distance-Based Algorithms

In the presented research work several K-NN versions are presented. Some of them attempt to minimize some of the K-NN weaknesses, whereas others offer an innovative extension.

Some of the versions are focused on attempting to select the best K value dynamically. Another version tries to reduce the influence of the outliers in K-NN. Finally another version gives to all existing classes the chance to take part in the final decision.

Summarizing the publication related to K-NN area:

- I. Mendialdua, N. Oses, B. Sierra, E. Lazkano. "Positive Predictive Value based dynamic K-Nearest Neighbor", In Proceedings of the 2012 Knowledge-Based Intelligent Information & Engineering Systems. 2012.
- B. Sierra, E. Lazkano, I. Irigoien, E. Jauregi, I. Mendialdua. "K Nearest Neighbor Equality: Giving equal chance to all existing classes", Information Sciences. 2011.

- I. Mendialdua, B. Sierra, E. Lazkano, I. Irigoien, E. Jauregi. "Decreasing K Nearest Neighbor". In Proceedings of the 2011 Conferencia de la Asociación Española para la Inteligencia Artificial. 2011.
- I. Mendialdua, B. Sierra, E. Lazkano, I. Irigoien, E. Jauregi. "Surrounding Influenced K-Nearest Neighbors: A New Distance Based Classifier", In Proceedings of the 2010 Advanced Data Mining and Applications. 2010

4 Multi-Classifiers

In the previous chapters we describe several paradigms widely used in the field of Supervised Classification which provide the researcher important data mining tools for data analysis. These include Decision Trees, Distance Based Algorithms or Bayesian Classifiers.

The first considerations when we have to deal with a classification problem consists of the following :

- Apply different algorithms to the problem.
- Evaluate the results through a validation technique.
- Select the algorithm that provides the best result.

However, with selecting the suitable combination of the classifiers, it can be possible to obtain better results than selecting only the classifier with the best result [70]. But in order to achieve this result, the combined classifiers must satisfy two conditions: each individual classifier has to be accurate and the classifiers to be combined must be diverse among them [51].

- An accurate classifier is the classifier that obtains better accuracy for the new cases than the expected accuracy of a classifier that guesses randomly.
- Two classifiers are diverse when they make different errors in a set of new individuals.

Table 4.1 illustrates an example of why these two conditions improve the hit rate. The example shows the results obtained by 3 diverse and accurate classifiers for 10 instances ($x_1 - x_{10}$). Each classifier obtains 60% hit rate, but it can be seen that by combining them it is possible to increment the accuracy to 80%.

A more comprehensive account of the reasons why the classifiers combinations outperforms the single ones is stated by Dietterich in [26]. He gives 3 reasons: statistical, computational and representational.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	Accuracy
Classifier1	✓	✓	✓	✓	✓	✓	×	×	×	×	60%
Classifier2	×	×	×	×	✓	✓	✓	✓	✓	✓	60%
Classifier3	✓	✓	✓	×	×	×	×	✓	✓	✓	60%
Vote	✓	✓	✓	×	✓	✓	×	✓	✓	✓	80%

Table 4.1: Improving the accuracy combining 3 diverse classifiers

- Statistical: there may be several accurate classifiers for a problem. Combining these accurate classifiers, the algorithm can average their votes and reduce the risk of choosing the wrong classifier.
- Computational: some classification algorithms use local search methods, which can lead to local optima. A classifier combination can expand the search space, and offer different local optima which may provide a better approximation to the true unknown function.
- Representational: it is possible that none of the single classifiers approximates to the true function. Nevertheless, the combination of classifiers might be able to get a more accurate function.

Commonly these combinations of classifiers are known as Classifier Ensembles or Multiple Classifier Combinations. Two main approaches for the design of Classifier Ensembles are defined in the literature: Classifier Fusion and Classifier Selection. In the latter, each ensemble member is supposed to have knowledge about the whole feature space; in the former, each ensemble member is supposed to know well a part of the feature space and be responsible for this subspace. Therefore, in the fusion approach, combiners such as the majority vote are applied, whereas in the selection approach, a classifier is selected to classify the new unlabelled instance.

4.1 Classifier Fusion

The motivation behind Classifier Fusion methods is to outperform individual classifiers by training several models on each problem and then combine their predictions by voting or by other methods. In the next sections some of the most popular Classifier Fusion strategies are described:

4.1.1 Bagging

Bagging is proposed by Breimann [8] and it is based on the concepts of bootstrapping and aggregation, hence its name Bootstrap AGGregatING. The idea of Bagging is to combine, with a simple vote, the results of various classifiers built on bootstrap replicates of the training set.

As we have seen, the combination of classifiers requires that the individual components are diverse from each other. Thus, when Bagging is applied the selected base classifier should be unstable. An unstable classifier is a classifier whose output is strongly affected by small changes in the training set. Decision Trees are an example of unstable classifiers and that is why they are one of the most used base classifiers in Bagging.

4.1.2 Random Forests

Random Forests are a variant of Bagging and are also proposed by Breiman [9]. The new training sets are created as in Bagging but the Decision Trees are built in a different way. In each node a random subset of features is selected and only those features are considered to split the node. The tree is expanded without pruning.

4.1.3 Boosting

Boosting attempts to turn a weak classifier in a considerably stronger classifier. While in Bagging all individual classifiers are built simultaneously, in Boosting they are built sequentially, based on the results of the previous ones. In each iteration, the new classifier assigns higher weight to those instances wrongly classified by the previous classifier. Adaboost [37] is the most popular Boosting version.

4.1.4 Stacked Generalization

Stacked Generalization is a well-known ensemble approach and it is also called Stacking [108, 119]. While ensemble strategies such as Bagging or Boosting obtain the final decision after a vote among the predictions of the individual classifiers, Stacking uses another individual classifier to perform the combination of the predictions in order to detect patters and improve the obtained accuracy.

Stacking is divided into two levels: in the level-0 each individual classifier makes a prediction independently, and in the level-1 these predictions are treated as the input values of another classifier, known as meta-classifier, which returns the final decision. Figure 4.1 illustrates an example of Stacking.

The data for training the meta-classifier is obtained after a validation process, where the outputs of the classifiers in level-0 are taken as attributes and the class is the true class of the examples.

Ting and Witten [113] propose to extend Stacked Generalization using class probability distributions of the original classifiers. Moreover, they propose to use the Multi-Response Linear Regression (MRLR) as meta-classifier.

Seewald [102] shows that this new version works correctly for two-class problems, while it performs worse for multi-class problems. In order to solve this problem, he introduces a new method called StackingC where, for each class separately, a meta training set is created with the class probabilities associated with the class. In this case, he also uses MRLR as meta-classifier.

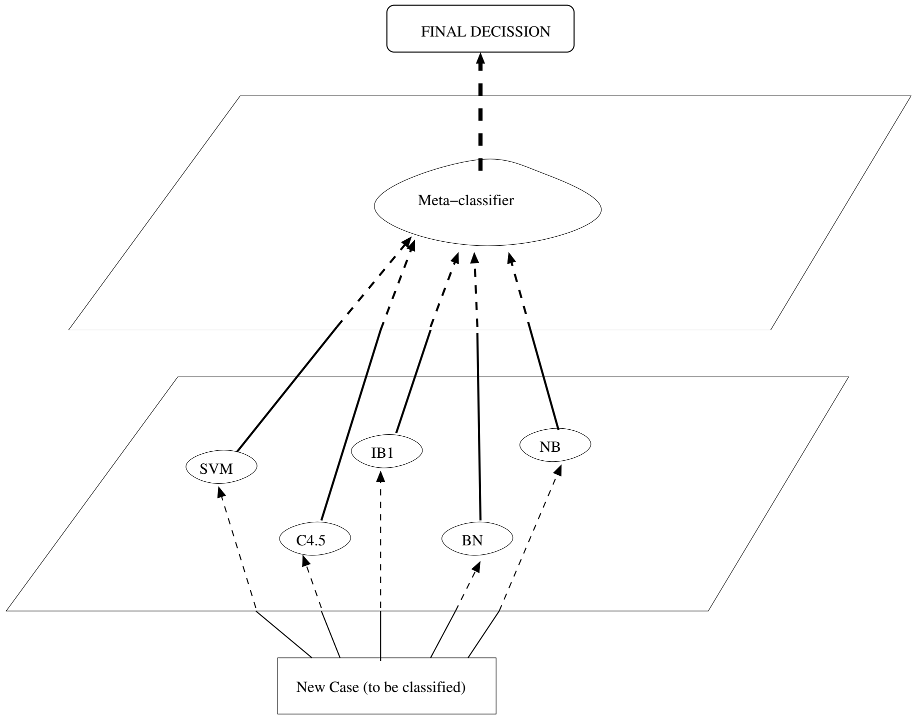


Fig. 4.1: Example of Stacking

On the other hand, there are some works that attempt to improve the performance of Stacked Generalization by selecting the best classifiers using different evolutionary computation strategies. Chen [12, 13], proposes a new ensemble construction method which applies Ant Colony Optimization (ACO) in the Stacked Generalization ensemble construction process to generate domain-specific configurations. Shunmugapriya and Kanmani [106], use Artificial Bee Colony(ABC) Algorithm as a meta-heuristic search algorithm to obtain a suitable Stacked Generalization model. To this end, two versions of the ABC algorithm are used. Ledezma et al. [73] use the genetic algorithms and show that selecting the right classifiers, their parameters and the meta-classifier is a critical issue.

4.2 Classifier Selection

The Classifier Selection strategies rely on the assumption that each classifier is an expert in different areas. To do so they divide the feature space into several local regions, and try to determine which classifier is the best in each one. Classifier selection techniques may be divided into two groups: static and dynamic.

4.2.1 Static Classifier Selection (SCS)

In the Static Classifier Selection (SCS) strategies, the local regions and the selected classifier for each of those are statically defined in the validation phase. An example can be found on [69] where the training data is clustered into K clusters and then the most successful classifier for each cluster is selected.

4.2.2 Dynamic Classifier Selection (DCS)

Dynamic Classifier Selection (DCS) strategies have received more attention in the literature. Instead of selecting the classifiers in the validation step, as SCS do, DCS strategies select the classifiers dynamically in the classification phase considering the characteristics of the sample to be classified.

The first Dynamic Classification approaches are introduced by Woods [120] and are based on the K-NN algorithm. He proposes two methods: Overall Local Accuracy (OLA) and Local Class Accuracy (LCA). OLA method obtains the K nearest neighbors of the new instance to be classified. It estimates each classifiers' accuracy for these K neighbors and the most accurate classifier is selected to classify the new instance. LCA method is similar to OLA but it considers the classifier's predicted class to calculate its local accuracy. In other words, if classifier C_j predicts the class θ_i for the new instance, the local accuracy of C_j is considered in the following sense: a confidence level is given to C_j which takes into account the percentage, among the K nearest neighbors, that have been correctly assigned to this θ_i class. The classifier which obtains the highest local accuracy is the selected one, thus, the class predicted by this classifier is the chosen one.

Smith [109] introduces an immediate extension of OLA applying the Distance Weighted K-NN (DW-OLA). In this way, each of the K neighbors receives a weight depending on their distance to the unknown instance where the closest neighbors have more influence in the classifier selection decision. Giacinto and Roli [49] also extend Woods's work proposing two new apporaches: A Priori and A Posteriori. A Priori strategy is similar to DW-OLA, but the selection condition is changed taking into account the posterior probability of each classifier in the K neighbors. The second proposed approach, A Posteriori, is a mixture of LCA with A Priori.

Other approaches use paradigms which are different to K-NN to assign the local regions; Liu and Yuan [77] propose to use clustering. Firstly, for each base classifier the feature space is divided into several clusters. Secondly, the unknown sample is assigned to a cluster for each classifier. And finally, the classifier of the most accurate cluster is selected to classify the unknown sample.

Recently, the DCS method has been extended to Dynamic Ensemble Selection (DES): instead of finding the most suitable classifier, the most suitable ensemble for each sample is selected. These methods can be considered as a combination of Classifier Selection and Classifier Fusion methods. Ko et al. [63] present 4 new dynamic selection schemes that explore the properties of the oracle concept. On the other hand, Dos Santos et al. [28] propose a two-step DES method: in the first step, highly accurate candidates ensembles are selected; in the second step, for each test sample, the ensemble with the largest confidence level is selected among those ensembles. In a further work Cavalin et al. [10] extend the previous work and adapt it to Dynamic Multistage Organization strategy.

4.3 Contributions on Multi-Classifiers

A new contribution of Multi-Classifiers is presented in this work. The main idea is to select the best subset of classifiers in each problem within Stacked Generalization Multi-Classifier. The following publication has been obtained:

- I. Mendialdua, A. Arruti, E. Jauregi, E. Lazkano, B. Sierra. "Classifier Subset Selection to construct multi-classifiers by means of estimation of distribution algorithms". Neurocomputing, 2015.

5 Class Binarization

As we have introduced previously, there exist different types of classifiers. For some of them, such as SVM, it is much easier to build a classifier to distinguish just between two classes. However, many real world problems are multi-class problems, i.e. $K > 2$, K being the number of classes. In view of that, there are some techniques, known as Class Binarization techniques, which divide the original multi-class problem into many binary classification problems. Usually the SVM algorithm is used as base classifier in all the binary sub-problems, but due to the good results obtained, the use of these strategies has been extended to other base classifiers, such as Ripper [40] or C4.5 [27]. In recent years, the Class Binarization strategies have been receiving more attention in the literature, and one indicative of that is that recently several reviews have been published [41, 46, 81]. Moreover, it has been applied successfully in different kinds of problems: ranking [59], computer vision [96], fingerprint identification [57], OCR [22], natural language processing [48, 116]....

It is usual to consider the Class Binarization methods as part of the Classifier Combination strategies. Although it is true that the Class Binarization strategies are ensembles, they follow different philosophies. While the Class Binarization strategies aim to solve multi-class problem by dividing the original into several binary sub-problems, the classical Classifier Combination strategies attempt to improve a single classifier by creating several new classifiers and combining them in order to obtain a new Multi-Classifier that outperforms all of them. Moreover, in the Multi-Classifiers each classifier is able to return any of the classes, while in Class Binarization strategies each sub-problem returns 0/1 results; this fact has allowed different output combination models.

5.1 Structure

Class-Binarization is composed of two steps: decomposition and combination.

In the decomposition step the original problem is divided into several binary sub-problems. The most popular strategies consist of grouping classes, in this way each binary classifier compares two groups of classes between them. Commonly, a so-called code-matrix is used to represent how the classes are grouped.

Figure 5.1 shows a code-matrix example, where 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. In Figure 5.1 how a 5-class problem $\{\theta_1,\theta_2,\theta_3,\theta_4,\theta_5\}$ is decomposed into 5 binary problems $\{f_1,f_2,f_3,f_4,f_5\}$ can be seen. For instance, it can be seen that the classification sub-problem 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 in θ_3 and θ_5 in class -1. So the classifier constructed to deal with f_1 classification sub-problem aims at distinguishing θ_1 and θ_2 classes on the one hand versus θ_3 and θ_5 on the other hand, while the cases that belong to θ_4 are not considered.

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

Fig. 5.1: Example of a code-matrix

In the classification step, each binary classifier returns a prediction. So the combination step consists of combining these predictions. Therefore, it is crucial to select a proper combination of the outputs, since depending on this decision the final decision can vary.

Different decomposition strategies have been developed. Two of the most popular are One-Vs-All and One-Vs-One, which are described below.

5.2 One-Vs-All (OVA)

OVA decomposition scheme divides a K class multi-class problem, $\theta_1, ..., \theta_K$, into K two-class problems, where each binary problem discriminates one class from the others.

In Figure 5.2a OVA's code-matrix for 4 classes is shown: in each classification problem one class is represented as the positive class while all the other 3 classes are represented as the negative-class.

As one class is compared with all the other classes, most of the binary sub-problems are unbalanced. It is known that one of the drawbacks of an unbalanced problem is the underestimation for the minority classes, thereby the most represented class is selected in most cases. In view of that, in OVA it is very common that all sub-problems return a class-negative prediction, hence, ties are usual in the final decision when the majority vote is used. Thus, it is more efficient to use the confidence level of each classifier to decide the final output. The class with the highest confidence is the selected one.

OVA is introduced by Anand et al. [3]. Among the Class Binarization strategies, OVA is that which has received less attention in the literature, and there are not many aggregations. Polat and Günes [89] propose to use OVA with C4.5 base classifier. Yang et al. [123] propose to integrate Decision Trees in OVA to order the sequence of classifiers. Hong et al. [57] introduce a similar idea but they integrate Naive Bayes in OVA; besides, the classifier sequence is obtained dynamically. On the other hand, Kumar and Gopal [68] present a method where they reduce the number of samples of the classifier discarding the instances that are located out of an established region.

5.3 One-Vs-One (OVO)

OVO decomposition scheme, also called Pairwise Classification, divides a K class multi-class problem, $\theta_1, ..., \theta_K$, into $K(K-1)/2$ two-class sub-problems. In each sub-problem a classifier is learned 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.

In Figure 5.2b OVO's code-matrix for 4 classes is presented: in each column one class is represented as +1 class, another one is represented as -1 and the remaining 2 classes are represented as 0.

$$\begin{pmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \end{pmatrix}$$

(a) One-Vs-All

$$\begin{pmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

(b) One-Vs-One

Fig. 5.2: OVA and OVO code-matrix

There are several strategies to combine the outputs. The simplest way is to use the majority vote strategy [38, 40] also called as Max-Wins; the most voted class is the selected one. An immediate extension is the Weighted Voting (WV) to use the confidence level of each binary problem as a vote. Its robustness has been shown in [60]. Hastie and Tibshirani [53] propose a new method called Pairwise Coupling (PC). The aim of the method is to find the best approximation of the class posterior probabilities given the posterior probabilities of the pairwise sub-problems. To do so, they transform the problem into an iterative one where they try to minimize the average weighted Kullback-Leibler divergence between the obtained pairwise estimates and the true pairwise probability values. Wu et al. [121] also estimate the posterior probabilities of each class, but the optimization formulation is different than the one presented in [53]. Jelonek and Stefanowsky [62] also extend [53] adding a credibility coefficient. Krzyśko and Wolyński [67] introduce several variations to combine the outputs of the sub-problems. The use of Belief Functions also has been proposed by Quost et al. [93]. Savicky and Fürnkranz [101] propose to replace the voting procedure with Stacked Generalization.

OVO has several drawbacks; 3 of the main disadvantages of OVO are the followings:

1. Unclassifiable regions: it is possible that each binary classifier votes for a different class, hence there is no winner. Thus, some tie-breaking technique is needed.
2. Number of classifiers: compared with OVA, it can be seen that OVO creates more sub-problems. Moreover, the disadvantage of having so many sub-problems is that most of them are irrelevant and they are forced to give wrong answers for many instances, because each classifier must assign every pattern to one of two classes. If a pattern belongs to class i, all the classifiers that are not trained to differentiate this class will cast wrong votes. However, OVO uses fewer examples in each sub-problem and, thus, has more freedom for fitting a decision boundary between the two classes.
3. Weak classifiers: the classical way is to select the optimal classifier for the dataset as base classifier. Therefore, all the sub-problems are classified with this classifier. As there are too many sub-problems, it is possible that this base classifier has difficulties to distinguish between all of them, thus, the classifiers return wrong results. This raises the question – should the same base classifier be used on all sub-problems or should sub-problems be tuned independently?

Several proposals have been developed in the literature in order to solve these problems.

In order to resolve the unclassifiable regions Platt et al. [88] publish a new combination proposal called Decision Directed Acyclic Graph (DDAG). DDAG builds a rooted binary acyclic graph where in each node a classifier discriminates between two classes. The final answer is the class assigned by the leaf node. Liu et al. [76] introduce a tie-breaking technique, in which OVO is applied using only the examples in the unclassifiable region.

On the other hand, other approaches try to reduce the number of binary classifiers in OVO. To do so, some proposals suggest using the Dynamic Classifier Selection [4, 42]. Park and Fürnkranz [86] propose to count the number of votes that a class does not received. When the following classes get more no-votes, further evaluations with this class are ignored because it is impossible to achieve higher total voting. Other authors propose the use of hierarchical structure. Fei and Liu [35] present a new architecture called Binary Tree of SVM (BTS). BTS is a binary tree where in each node two classes are distinguished. The main idea of BTS is to use the separating plane for these two classes, and also to distinguish other classes. Chen et al. [11] introduce a new BTS version where they try to select the binary SVM with the fewest number of separating lines. Following the same idea, to reduce the number of classifiers, the hierarchical structure has been extended to other Class Binarization strategies [47, 80, 82, 90].

Finally, other approaches try to solve the weak classifier problem using each sub-problem independently. On the one hand, some proposals focus on attempting to select the best base classifier in each sub-problem [110]. On the other hand, other approaches try to select the best hyper-parameters of SVM in each sub-problem. Because of the high number of possible values of the hyper-parameters, most of these works use evolutionary algorithms. Lebrun et al. [72] and Liepert [75] propose the use of Genetic Algorithms while Souza et al. [20] propose the use of Particle Swarm Optimization. The results obtained by these four works are contradictory, since two of them consider that the independent tune of the sub-problems is better while the other two consider that there is no significant difference.

Lorena and Carvalho [79] consider that none of the mentioned works performed a rigorous statistical analysis. Thus, they investigated the use of Genetic Algorithms to automatically tune the parameters of each binary SVM. They conclude that the use of the same parameter values in all binary SVM is sufficient to obtain good results.

In his PhD thesis, Reid [94] conclude that, although it is better to use the same base classifier for all the sub-problems when decision boundaries of the sub-problems have similar shapes, in the cases where the decision boundaries have a different shape it is better to treat sub-problems independently.

5.3.1 Combining OVA and OVO

Some authors propose new approaches based on the combination of OVA and OVO. On the one hand, Moreira and Mayoraz [85] propose to apply OVO taking into account the probability that the new example belonged to each pair of classes. This probability is obtained following the OVA idea: creating a classifier that distinguishes between the two classes and the rest of the classes. On the other hand, García-Pedrajas and Ortiz-Boyer [45] and Ko and Byun [64] present a very similar idea to combine OVA and OVO. In an interesting motivation section, García-Pedrajas and Ortiz-Boyer [45] show that in the majority of the cases the correct class is between the two largest confident outputs of OVA. Thus, in their new method they obtain the two classes with the highest confidence level in OVA first. After that, OVO is applied and a classifier is built taking into account only the cases that belong to these two classes.

5.4 Error Correcting Output Codes (ECOC)

Error Correcting Output Codes (ECOC) framework is introduced by Dietterich and Bakiri [27] although its origins can be found in the information transmission [104]. The original ECOC method requires all classes to appear in each sub-problem; moreover, the code-matrix has more columns (classifiers) than required. Those additional columns introduce a redundancy which provide the system the capability to recover from classification errors committed. Furthermore, the better the separability of the rows and the better the separability of the columns, the better the error correcting capability obtained. To take the final decision, the authors propose to create an output-vector which contains the results of each sub-problem. This output-vector is compared with the row of each code-matrix and the class of the row with the minimum Hamming distance is assigned to the new instance.

In the ECOC proposed by Dietterich and Bakiri [27], the "0" symbol in the code-matrix is not allowed. Thus, Allwein et al. [2] propose to extend ECOC scheme introducing the "0" symbol, allowing some classes to be ignored in the binary classifiers. Furthermore, they consider that only applying Hamming distance, essential information is lost. Thus, they introduce the Loss-Based decoding which considers the confidence level of the classifiers.

Several ECOC versions have been developed with different aims. Some of them try to outperform the outputs combination by proposing to use Euclidean Distance [32] or improving the Loss-Based decoding [33].

As there are too many possible code-matrix solutions, several authors have focused on reducing the number of sub-problems. Bautista et al. [5] introduce the minimal ECOC design, whereas other authors proposed ECOCs that can be converted into binary trees structured hierarchical classifier, where in each node a binary partition is made [90]. On the other hand, Pimenta and Gama [87] present a new strategy to define the best number of sub-problems for each multiclass problem.

5.5 Comparison

In several works different Class Binarization strategies have been compared. Some of them conclude that OVO is significantly better than OVA [40, 58]. However, Rifkin and Klautau [95] suggest that when the binary classifiers are well-tuned, OVA performs as well as the other strategies. Recently, two empirical studies have appeared concerning this question [41, 46]. Galar et al. [41] compare different OVO and OVA strategies. While García-Pedrajas [46] 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.

5.6 Contribution on Class Binarization

This work is focused on Class Binarization strategy, especially in OVO method, with which the main contributions have been achieved. Three main contributions are presented. One of them reduces the number of sub-problems in OVO strategy. Another one is an OVA and OVO combination which selects a different base classifier in each sub-problem statically. And the last proposal selects the base classifier in each sub-problem dynamically.

With these works, the following has been published in international journal:

- A. Arruti, I. Mendialdua, B. Sierra, E. Lazkano, E. Jauregi. "$NewOne-Versus_{One}^{All}$ method: NOV@". Expert Systems with Applications, 2014.

And another two articles have been submitted to international journals:

- I. Mendialdua, G. Echegaray, I. Rodriguez, E. Lazkano, B. Sierra. "Undirected Cyclic Graph Based Multiclass Pair-wise Classifier: classifier number reduction maintaining accuracy", Neurocomputing.
- I. Mendialdua, J. M. Martínez-Otzeta, I. Rodriguez, T. Ruiz-Vazquez, B. Sierra. "Dynamic selection of the best base classifier in One versus One", Knowledge-Based Systems.

6

Contributions

The main contributions of this research work are briefly described in this section.

6.1 Contributions published in international journals

6.1.1 K Nearest Neighbor Equality: Giving equal chance to all existing classes

Authors: B. Sierra, E. Lazkano, I. Irigoien, E. Jauregi and I. Mendialdua.

Journal: Information Sciences. **(Q1)**

Year: 2011

An extension of the K-NN algorithm is presented in this paper. The new approach is called K Nearest Neighbor Equality (K-NNE), since all the classes take part in the final decision. When an instance has to be classified according to the K-NN algorithm, its K nearest neighbors are obtained and the most represented class among those neighbors is predicted for the new instance. However, if there are classes that are not represented among those neighbors, they are not taken into account. In order to make all the classes participate, K-NNE finds the K nearest neighbors of each class for the new instance. The mean distance of the K neighbors of each class is calculated and the class with the minimum mean distance is predicted. This proposal is compared with the classical K-NN algorithm for different K values obtaining promising results. In order to verify this conclusion, the Wilcoxon test is applied, which confirms the statistical significance of the good performance of the proposal.

6.1.2 $New\ One\ Versus_{One}^{All}$ method: NOV@

Authors: A. Arruti, I. Mendialdua, B. Sierra, E. Lazkano and E. Jauregi.

Journal: Expert Systems with Applications. **(Q1)**

Year: 2014

In this paper, two new algorithms that combine OVA and OVO are presented. The description of OVA and OVO algorithms can be found in section 5. Usually OVA and OVO use the same base classifier for every sub-problem; however, some approaches treat each sub-problem independently and their aim is to find the best base classifier for each sub-problem. The presented approaches are an extension of these works. The first approach described in the paper is called OVA+OVO. OVA+OVO is a simple combination of OVA and OVO: the multi-class problem is decomposed following OVA and OVO strategies and the outputs of the sub-problems are combined using the majority vote. An empirical study is carried out, where the new proposal is compared with other state-of-the-art algorithms. Due to the bad results obtained by OVA, its results are analyzed and therefore the second approach called $NewOne - Versus_{One}^{All}$ (NOV@) is proposed. NOV@ is also a combination of OVA and OVO strategies: OVA is applied and the majority vote is used for the OVA outputs. If there are ties, OVO is applied considering the tie-classes. Once again, an empirical study is carried out and this time NOV@ is incorporated to the previous comparison. OVO+OVA and NOV@ show a good performance and the Iman-Davenport test confirms the statistical significance of this behaviour. Furthermore, the computational load of the algorithm is analyzed and it is shown that NOV@ needs fewer sub-problems than OVO to achieve better results.

6.1.3 Classifier Subset Selection to Construct Multi-Classifiers by means of Estimation of Distribution Algorithms

Authors: I. Mendialdua, A. Arruti, B. Sierra, E. Lazkano and E. Jauregi.

Journal: Neurocomputing. **(Q1)**

Year: 2015

In this paper a new algorithm that tries to select the best subset of classifiers for each database is presented. Hence, the new approach is called Classifier Subset Selection (CSS). To combine the output of the selected classifiers, the well-known Multi-Classifier technique called Stacked Generalization is used. In this work, a set of 10 single classifiers are provided initially and in the validation phase the goal is to select the most diverse and accurate subset of them; to do so, an evolutionary approach called Estimation of Bayesian Network Algorithm (EBNA) is used. This proposal is compared with other Multi-Classifiers obtained from the state-of-the-art. The obtained results are very good and the statistical test applied confirms the robustness of the proposal.

6.2 Contributions submitted to international journals

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

Authors: I. Mendialdua, G. Echegaray, I. Rodriguez, E. Lazkano and B. Sierra.

Journal: Neurocomputing. **(Q1)**

The new approach presented in this paper is called Decision Undirected Cyclic Graph (DUCG) and its aim is to reduce the number of sub-problems in OVO. To do so, an undirected cyclic graph is created where each node represents a class and has degree 2. Instead of comparing all the classes between them, each class is compared with the classes that are connected in the graph. To create the graph and try to find the best class comparisons, an evolutionary computation approach from the state-of-the-art called Edge Histogram-Based Sampling Algorithm (EHBSA) is used. An empirical comparison with other state-of-the-art algorithms is carried out over 4 different classifiers. The obtained results show that OVO and DUCG are the most robust algorithms, although it is worth mentioning that DUCG needs fewer base classifiers than OVO.

6.2.2 Dynamic selection of the best base classier in One versus One

Authors: I. Mendialdua, J. M. Martínez-Otzeta, I. Rodriguez, T. Ruiz-Vazquez and B. Sierra.

Journal: Knowledge-Based Systems. **(Q1)**

A new approach is presented that combines the research lines developed in this thesis project: Class Binarization, Multi-Classifiers and K-NN. The aim of the proposal is to select the best base classifier for each sub-problem of OVO dynamically for each unseen instance. The new proposal is called DYNOVO. Several variations of the proposal are shown. Two of them use two Dynamic Classifier Selection techniques from the state-of-the-art: Overall Local Accuracy (OLA) and Distance Weighted Overall Local Accuracy (DW-OLA). Both methods use K-NN algorithm to obtain the local region of the unseen instance. Another two variations of DYNOVO are also proposed: when OLA or DW-OLA are applied, instead of using K-NN, it is proposed to use K Nearest Neighbor Equality (K-NNE) algorithm. The proposal is compared with other state-of-the-art methods and it obtains very good results, especially when OVO, DW-OLA and K-NNE are combined. The Iman-Davenport test confirms the statistical significance of the good performance of DYNOVO. Moreover, the computational load of DYNOVO is also analyzed.

6.3 Contributions published in international conferences

6.3.1 Surrounding Influenced K-Nearest Neighbors: A New Distance Based Classifier

Authors: I. Mendialdua, B. Sierra, E. Lazkano, I. Irigoien and E. Jauregi.

Conference: International Conference on Advanced Data Mining and Applications. **(CoreB)**

Year: 2010

A new version of the K-NN is proposed in this paper. The aim of the algorithm is to reduce the influence of the outliers in K-NN algorithm. To do so, the limits of the K neighbors are extended obtaining their I nearest neighbors, where I is a new parameter introduced. The proposed method is compared with the original K-NN algorithm obtaining promising results for two-class problems.

6.3.2 Positive Predictive Value based dynamic K-Nearest Neighbor

Authors: I. Mendialdua, N. Oses, B. Sierra and E. Lazkano.

Conference: International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. **(CoreB)**

Year: 2012

In this paper a new version of K-NN is introduced whose goal is to select the most reliable K for each test instance. When a test instance to be classified arrives, it is classified for different K values. Observing for each K how many votes the winning class receives, the K with the most reliable prediction is selected. To calculate the reliability, the Positive Predicted Values (PPV) obtained in a validation process are used. The new algorithm is compared with K-NN and gets interesting results.

6.3.3 Combination of one-vs-one and one-vs-all using different base classifiers.

Authors: I. Mendialdua, B. Sierra, E. Lazkano and E. Jauregi.

Conference: The European Research Consortium for Informatics and Mathematics. **(CoreB)**

Year: 2012

It is worth mentioning that in ERCIM only the abstract is presented and not the full paper. Therefore, in the part where the contributions are presented there is no contribution with this title.

A preliminary version of the article "$NewOneVersus_{One}^{All}$ method: NOV@" is presented in this conference. To be more precise, the first combination of OVA and OVO called OVA+OVO is presented. In order to validate the performance of the OVA+OVO, an empirical study has been done where the new method has been compared with OVA and OVO. Experimental results show that the new method gets promising results.

6.4 Contributions published in national conferences

6.4.1 Decreasing K Nearest Neighbor

Authors: I. Mendialdua, B. Sierra, E. Lazkano, I. Irigoien and E. Jauregi.

Conference: Conferencia de la Asociación Española para la Inteligencia Artificial

Year: 2011

In this paper a new version of the K-NN algorithm is proposed. K-NN sometimes makes the prediction without enough certainty. In order to solve this problem, a new algorithm that selects the value of the K parameter dynamically is presented. This method is called Decreasing K Nearest Neighbor (DK-NN). DK-NN starts assigning to the K parameter a high value, and this value decreases until the most voted class exceeds a percentage of votes. The aim is to assure that the final decision is made with enough confidence. The proposed method is compared with K-NN algorithm obtaining interesting results.

7 Conclusion

This work has been devoted to developing several contributions in the Supervised Classification area. To do so, besides combining several Supervised Classification strategies, other approaches of Artificial Intelligence such as Evolutionary Algorithms have also been used.

The contributions have been made in 3 research lines: Distance-Based algorithms, Multi-Classifiers and Class Binarization techniques.

Distance-Based Algorithms: the first contributions of this work are the proposal of several versions of the K-NN algorithm. The main objective of these versions is to try to solve some of the weaknesses of K-NN proposing innovative ideas. The main contribution among those versions is the K Nearest Neighbor Equality (K-NNE) algorithm, whose relevance is supported by a publication in a Q1 journal.

Multi-Classifiers: a novel extension of the well-known Stacked Generalization Multi-Classifier is introduced. The aim of the proposal is to select the best subset of classifiers.

Class Binarization: in the Class Binarization research line some of the most relevant works proposed in this thesis project can be found. A study of the Class Binarization strategies has been carried out, where the majority of works are focused on the strategy One versus One (OVO). On the one hand, it is focused on the reduction of the number of sub-problems in OVO. To do so, a new algorithm called Decision Undirected Cyclic Graph (DUCG) is proposed. On the other hand, novel algorithms that treat independently each sub-problem and tries to find the best classifier for each sub-problem have been tested. Two proposals have been developed in this way. The first one is called $NewOne - Versus_{One}^{All}$ (NOV@) and it is a combination of OVO and One versus All (OVA). The second one is called DYNOVO and its aim is to select for each test instance the best base classifier in each sub-problem. Moreover, DYNOVO can be seen as the union of the three research lines followed in this thesis project, since it combines K-NN algorithm, Dynamic Classifier Selection strategies and OVO.

7.1 Future work

The contributions presented in this thesis project cover several gaps in the areas mentioned above; however, there are several lines of work that can be studied in the future. The most direct one is the extension of these methods to real life problems, especially to those problems that the research group I belong to is dealing with. Another option is to continue improving the proposed approaches and to continue trying to solve some of the weaknesses that are found in the state-of-the-art algorithms. In this section some of the future works for each of the research lines are presented:

Distance-Based Algorithms: given the good synergy observed between K-NNE and Dynamic Classifiers Selection strategies for binary problems, it would be interesting to analyze this combination in multi-class problems.

Multi-Classifiers: on the one hand, we are working in the next version of Classifier Subset Selection (CSS), where the objective is to try to select the best subset of classifiers for each test instance dynamically. On the other hand, given the interesting results obtained by the proposed Classifier Subset Selection algorithm, the aim is to apply the proposal in real problems. Two options are treated in this sense: the first one is to apply in Natural Language Problems dependency parsing problems: the objective would be to predict the dependency relations among the words of a sentence. The second one is to apply in Emotions Recognition.

Class Binarization: related to the Class Binarization strategies, there are also several promising work lines. One of them could be working on another version of OVO with the aim of reducing the number of sub-problems applying One-Class classification strategies. The One-Class classifiers decide if the unseen instance belongs to the same class of the training set instances or to another unknown class. Thus, the idea is to classify the unseen instance with the One-Class classifier in each sub-problem. If the classifier predicts that the instance belongs to the same class as the training data, the output of the sub-problem is considered, otherwise the sub-problem is discarded. Another idea is to combine the proposed CSS method with OVO, where in each sub-problem the CSS is used as base classifier.

References

[1] Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine learning*, 6(1):37–66.

[2] Allwein, E. L., Schapire, R. E., and Singer, Y. (2001). Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141.

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

[4] Bagheri, M., Gao, Q., and Escalera, S. (2012). Efficient pairwise classification using local cross off strategy. In *Proceedings of the 25th Canadian conference on Advances in Artificial Intelligence*, pages 25–36.

[5] Bautista, M. Á., Escalera, S., Baró, X., Radeva, P., Vitriá, J., and Pujol, O. (2012). Minimal design of error-correcting output codes. *Pattern Recognition Letters*, 33(6):693–702.

[6] Ben-David, A. (2007). A lot of randomness is hiding in accuracy. *Engineering Applications of Artificial Intelligence*, 20(7):875–885.

[7] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152.

[8] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

[9] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[10] Cavalin, P. R., Sabourin, R., and Suen, C. Y. (2013). Dynamic selection approaches for multiple classifier systems. *Neural Computing and Applications*, 22(3-4):673–688.

[11] Chen, J., Wang, C., and Wang, R. (2009). Adaptive binary tree for fast svm multiclass classification. *Neurocomputing*, 72(13):3370–3375.

[12] Chen, Y. and Wong, M. L. (2011). Optimizing stacking ensemble by an ant colony optimization approach. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 7–8.

[13] Chen, Y. and Wong, M.-L. (2012). Applying ant colony optimization in configuring stacking ensemble. In *Joint 6th International Conference on Soft Computing and Intelligent Systems and 13th International Symposium on Advanced Intelligent Systems*, pages 2111–2116.

[14] Clark, P. and Niblett, T. (1989). The cn2 induction algorithm. *Machine learning*, 3(4):261–283.

[15] Cleary, J. G. and Trigg, L. E. (1995). K*: an instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine Learning*, pages 108–114.

[16] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

[17] Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123.

[18] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

[19] Dasarathy, B. V. (1991). *Nearest Neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press.

[20] de Souza, B., de Carvalho, A., Calvo, R., and Ishii, R. (2006). Multiclass svm model selection using particle swarm optimization. In *Proceedings of the 6th International Conference on Hybrid Intelligent Systems*, pages 31–31.

[21] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

[22] Deng, H., Stathopoulos, G., and Suen, C. Y. (2009). Error-correcting output coding for the convolutional neural network for optical character recognition. In *Proceedings of the 10th International Conference on Document Analysis and Recognition*, pages 581–585.

[23] Denoeux, T. (1995). A k-nearest neighbor classification rule based on dempster-shafer theory. *IEEE Transactions on Systems, Man and Cybernetics*, 25(5):804–813.

[24] Devroye, L. and Wagner, T. J. (1979). Distribution-free performance bounds with the resubstitution error estimate. *IEEE Transactions on Information Theory*, 25(2):208–210.

[25] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

[26] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857, pages 1–15. Springer.

[27] Dietterich, T. G. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286.

[28] Dos Santos, E. M., Sabourin, R., and Maupin, P. (2008). A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern Recognition*, 41(10):2993–3009.

[29] Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(4):325–327.

[30] Efron, B. (1983). Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American Statistical Association*, 78(382):316–331.

[31] Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: the .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560.

[32] Escalera, S., Pujol, O., and Radeva, P. (2006). Ecoc-one: A novel coding and decoding strategy. In *Proceedings of the 18th International Conference on Pattern Recognition*, volume 3, pages 578–581.

[33] Escalera, S., Pujol, O., and Radeva, P. (2008). Loss-weighted decoding for error-correcting output coding. In *Proceesdings of the International Conference on Computer Vision Theory and Applications*, pages 117–122.

[34] Fayed, H. A. and Atiya, A. F. (2009). A novel template reduction approach for the-nearest neighbor method. *IEEE Transactions on Neural Networks*, 20(5):890–896.

[35] Fei, B. and Liu, J. (2006). Binary tree of svm: a new fast multiclass training and classification algorithm. *IEEE Transactions on Neural Networks*, 17(3):696–704.

[36] Fix, E. and Hodges Jr, J. L. (1951). Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document.

[37] Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(1612):771–780.

[38] Friedman, J. (1996). Another approach to polychotomous classifcation. Technical report, Technical report, Stanford University, Department of Statistics.

[39] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

[40] Fürnkranz, J. (2002). Round robin classification. *The Journal of Machine Learning Research*, 2:721–747.

[41] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., and 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(8):1761–1776.

[42] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., and Herrera, F. (2013). Dynamic classifier selection for one-vs-one strategy: Avoiding non-competent classifiers. *Pattern Recognition*, 46(12):3412–3424.

[43] García, S., Fernández, A., Luengo, J., and 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(10):2044–2064.

[44] García, S. and Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, pages 2677–2694.

[45] García-Pedrajas, N. and Ortiz-Boyer, D. (2006). Improving multiclass pattern recognition by the combination of two strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1001–1006.

[46] García-Pedrajas, N. and Ortiz-Boyer, D. (2011). An empirical study of binary classifier fusion methods for multiclass classification. *Information Fusion*, 12(2):111–130.

[47] Ghaffari, H. R. and Yazdi, H. S. (2013). Multiclass classifier based on boundary complexity. *Neural Computing and Applications*, 24(5):985–993.

[48] Ghani, R. (2000). Using error-correcting codes for text classification. In *Proceedings of the 17th International Conference on Machine Learning*, pages 303–310.

[49] Giacinto, G. and Roli, F. (1999). Methods for dynamic classifier selection. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 659–659.

[50] Han, E.-H., Karypis, G., and Kumar, V. (2001). Text categorization using weight adjusted k-nearest neighbor classification. *Advances in Knowledge Discovery and Data Mining*, pages 53–65.

[51] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

[52] Hastie, T. and Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616.

[53] Hastie, T. and Tibshirani, R. (1998). Classification by pairwise coupling. *The annals of statistics*, 26(2):451–471.

[54] Hellman, M. E. (1970). The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185.

[55] Hills, M. (1966). Allocation rules and their error rates. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–31.

[56] Holte, R. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91.

[57] Hong, J.-H., Min, J.-K., Cho, U.-K., and Cho, S.-B. (2008). Fingerprint classification using one-vs-all support vector machines dynamically ordered with naı¨ ve bayes classifiers. *Pattern Recognition*, 41(2):662–671.

[58] Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425.

[59] Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897–1916.

[60] Hüllermeier, E. and Vanderlooy, S. (2010). Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting. *Pattern Recognition*, 43(1):128–142.

[61] Iman, R. L. and Davenport, J. M. (1980). Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595.

[62] Jelonek, J. and Stefanowski, J. (1998). Experiments on solving multiclass learning problems by n 2-classifier. In *Proceedings of the 10th European Conference on Machine Learning*, pages 172–177.

[63] Ko, A. H., Sabourin, R., and Britto Jr, A. S. (2008). From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, 41(5):1718–1731.

[64] Ko, J. and Byun, H. (2003). Binary classifier fusion based on the basic decomposition methods. In *Proceedings of the 4th international conference on Multiple classifier systems*, pages 146–155. Springer.

[65] Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the 2nd International Conference on Knoledge Discovery and Data Mining*, pages 202–207.

[66] Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence*, volume 14, pages 1137–1145.

[67] Krzyśko, M. and Wołyński, W. (2009). New variants of pairwise classification. *European Journal of Operational Research*, 199(2):512–519.

[68] Kumar, M. A. and Gopal, M. (2011). Reduced one-against-all method for multiclass {SVM} classification. *Expert Systems with Applications*, 38(11):14238–14248.

[69] Kuncheva, L. I. (2000). Clustering-and-selection model for classifier combination. In *Proceedings of the 4th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, volume 1, pages 185–188.

[70] Kuncheva, L. I. (2004). *Combining pattern classifiers: methods and algorithms.* John Wiley and Sons, Inc.

[71] Larson, S. C. (1931). The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, 22(1):45–55.

[72] Lebrun, G., Lezoray, O., Charrier, C., and Cardot, H. (2007). An ea multi-model selection for svm multiclass schemes. In *Proceedings of the 9th international work conference on Artificial neural networks*, pages 260–267.

[73] Ledezma, A., Aler, R., Sanchs, A., and Borrajo, D. (2010). Ga-stacking: evolutionary stacked generalization. *Intelligent Data Analysis*, 14(1):89–119.

[74] Liaw, Y.-C., Leou, M.-L., and Wu, C.-M. (2010). Fast exact k nearest neighbors search using an orthogonal search tree. *Pattern Recognition*, 43(6):2351–2358.

[75] Liepert, M. (2003). Topological fields chunking for german with svms: optimizing svm-parameters with gas. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*.

[76] Liu, B., Hao, Z., and Yang, X. (2007). Nesting algorithm for multi-classification problems. *Soft Computing*, 11(4):383–389.

[77] Liu, R. and Yuan, B. (2001). Multiple classifiers combination by clustering and selection. *Information Fusion*, 2(3):163–168.

[78] Liu, Z.-g., Pan, Q., and Dezert, J. (2013). A new belief-based k-nearest neighbor classification method. *Pattern Recognition*, 46(3):834–844.

[79] Lorena, A. C. and De Carvalho, A. C. (2008). Evolutionary tuning of svm parameter values in multiclass problems. *Neurocomputing*, 71(16):3326–3334.

[80] Lorena, A. C. and de Carvalho, A. C. (2010). Building binary-tree-based multiclass classifiers using separability measures. *Neurocomputing*, 73(1618):2837–2845.

[81] Lorena, A. C., de Carvalho, A. C., and Gama, J. M. (2008). A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, 30(1-4):19–37.

[82] Madzarov, G. and Gjorgjevikj, D. (2009). Multi-class classification using support vector machines in decision tree architecture. In *IEEE EUROCON 2009*, pages 288–295.

[83] Martínez-Otzeta, J. M. and Sierra, B. (2004). Analysis of the iterated probabilistic weighted k nearest neighbor method, a new distance-based algorithm. In *Proceedings of the International Conference on Enterprise Information Systems*, pages 233–240.

[84] Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.

[85] Moreira, M. and Mayoraz, E. (1998). Improved pairwise coupling classification with correcting classifiers. In *Proceedings of the 10th European Conference on Machine Learning*, pages 160–171.

[86] Park, S.-H. and Fürnkranz, J. (2007). Efficient pairwise classification. In *Machine Learning: ECML 2007*, pages 658–665. Springer.

[87] Pimenta, E., Gama, J., and Carvalho, A. (2008). The dimension of ecocs for multiclass classification problems. *International Journal on Artificial Intelligence Tools*, 17(03):433–447.

[88] Platt, J. C., Cristianini, N., and Shawe-Taylor, J. (2000). Large margin dags for multiclass classification. *Advances in neural information processing systems*, 12(3):547–553.

[89] Polat, K. and Güneş, S. (2009). A novel hybrid intelligent method based on c4. 5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Systems with Applications*, 36(2):1587–1592.

[90] Pujol, O., Radeva, P., and Vitria, J. (2006). Discriminant ecoc: a heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1007–1012.

[91] Quinlan, R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[92] Quinlan, R. (1993). *C4.5: programs for machine learning.* Morgan Kaufmann Publishers, San Mateo, CA.

[93] Quost, B., Denoeux, T., and Masson, M.-H. (2007). Pairwise classifier combination using belief functions. *Pattern Recognition Letters*, 28(5):644–653.

[94] Reid, S. R. (2010). *Model combination in multiclass classification.* PhD thesis, University of Colorado.

[95] Rifkin, R. and Klautau, A. (2004). In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141.

[96] Rocha, A., Hauagge, D. C., Wainer, J., and Goldenstein, S. (2010). Automatic fruit and vegetable classification from images. *Computers and electronics in agriculture*, 70(1):96–104.

[97] Rodriguez, C., Boto, F., Soraluze, I., and Pérez, A. (2002). An incremental and hierarchical k-nn classifier for handwritten characters. In *Proceedings of the 16th International Conference on Pattern Recognition, 2002.*, volume 3, pages 98–101.

[98] Rodríguez, J. D., Pérez, A., and Lozano, J. A. (2013). A general framework for the statistical analysis of the sources of variance for classification error estimators. *Pattern Recognition*, 46(3):855–864.

[99] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

[100] Sáez, J. A., Derrac, J., Luengo, J., and Herrera, F. (2014). Statistical computation of feature weighting schemes through data estimation for nearest neighbor classifiers. *Pattern Recognition*, 47(12):3941–3948.

[101] Savicky, P. and Fürnkranz, J. (2003). Combining pairwise classifiers with stacking. In *Advances in Intelligent Data Analysis V*, Lecture Notes in Computer Science, pages 219–229. Springer.

[102] Seewald, A. (2002). How to make stacking better and faster while also taking care of an unknown weakness. In *Proceedings of the 19th International Conference on Machine Learning*, pages 554–561.

[103] Shaffer, J. P. (1986). Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831.

[104] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423.

[105] Shin, K., Abraham, A., and Han, S. Y. (2006). Improving knn text categorization by removing outliers from training set. *Computational Linguistics and Intelligent Text Processing*, pages 563–566.

[106] Shunmugapriya, P. and Kanmani, S. (2013). Optimization of stacking ensemble configurations through artificial bee colony algorithm. *Swarm and Evolutionary Computation*, 12:24–32.

[107] Sierra, B. and Lazkano, E. (2002). Probabilistic-weighted k nearest neighbor algorithm: a new approach for gene expression based classifica-

tion. In *Proceedings of the International Conference on Knowledge-Based Intelligent Information & Engineering Systems.*, pages 932–939.

[108] Sierra, B., Serrano, N., Larrañaga, P., Plasencia, E. J., Inza, I., Jiménez, J. J., Revuelta, P., and Mora, M. L. (2001). Using bayesian networks in the construction of a bi-level multi-classifier. a case study using intensive care unit patients data. *Artificial Intelligence in Medicine*, 22(3):233–248.

[109] Smits, P. C. (2002). Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection. *IEEE Transactions on Geoscience and Remote Sensing*, 40(4):801–813.

[110] Szepannek, G., Bischl, B., and Weihs, C. (2009). On the combination of locally optimal pairwise classifiers. *Engineering Applications of Artificial Intelligence*, 22(1):79–85.

[111] Tahir, M. A., Bouridane, A., and Kurugollu, F. (2007). Simultaneous feature selection and feature weighting using hybrid tabu search/k-nearest neighbor classifier. *Pattern Recognition Letters*, 28(4):438–446.

[112] Tan, S. (2005). Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667–671.

[113] Ting, K. M. and Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289.

[114] Wang, J., Neskovic, P., and Cooper, L. N. (2006). Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence. *Pattern Recognition*, 39(3):417–423.

[115] Wang, Q., Kulkarni, S. R., and Verdú, S. (2009). Divergence estimation for multidimensional densities via-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405.

[116] Wang, T.-Y. and Chiang, H.-M. (2009). One-against-one fuzzy support vector machine classifier: An approach to text categorization. *Expert Systems with Applications*, 36(6):10030–10034.

[117] Wettschereck, D. (1994). *A study of distance-based machine learning algorithms*. PhD thesis.

[118] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.

[119] Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

[120] Woods, K. (1997). Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410.

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

[122] Xie, Z., Hsu, W., Liu, Z., and Lee, M. L. (2002). Snnb: A selective neighborhood based naive bayes for lazy learning. In *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 104–114.

[123] Yang, X., Yu, Q., He, L., and Guo, T. (2013). The one-against-all partition based binary tree support vector machine algorithms for multi-class classification. *Neurocomputing*, 113:1–7.

Part II

Articles Related to Distance-Based Algorithms

8

K Nearest Neighbor Equality: Giving equal chance to all existing classes

Title: K Nearest Neighbor Equality: Giving equal chance to all existing classes

Authors: B. Sierra, E. Lazkano, I. Irigoien, E. Jauregi and I. Mendialdua.

Journal: Information Sciences.

Publisher: Elsevier

Year: 2011

DOI: http://dx.doi.org/10.1016/j.ins.2011.07.024

Information Sciences 181 (2011) 5158–5168

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

K Nearest Neighbor Equality: Giving equal chance to all existing classes

B. Sierra [*], E. Lazkano, I. Irigoien, E. Jauregi, I. Mendialdua

Department of Computer Science and Artificial Intelligence, University of the Basque Country, Spain

ARTICLE INFO

Article history:
Received 8 June 2009
Received in revised form 16 February 2011
Accepted 4 July 2011
Available online 23 July 2011

Keywords:
Nearest neighbor
Supervised classification
Machine Learning
Non-parametric pattern recognition

ABSTRACT

The nearest neighbor classification method assigns an unclassified point to the class of the nearest case of a set of previously classified points. This rule is independent of the underlying joint distribution of the sample points and their classifications. An extension to this approach is the *k*-NN method, in which the classification of the unclassified point is made by following a voting criteria within the *k* nearest points.

The method we present here extends the *k*-NN idea, searching in each class for the *k* nearest points to the unclassified point, and classifying it in the class which minimizes the mean distance between the unclassified point and the *k* nearest points within each class. As all classes can take part in the final selection process, we have called the new approach *k* Nearest Neighbor Equality (*k*-NNE).

Experimental results we obtained empirically show the suitability of the *k*-NNE algorithm, and its effectiveness suggests that it could be added to the current list of distance based classifiers.

1. Introduction

In a supervised classification problem based on a sample of *p*-variate observations $\mathbf{x}_1,\ldots,\mathbf{x}_n$ classified in $\theta_1,\ldots,\theta_M$ classes or populations, and given a new observation or case $\mathbf{x}$, the aim is to classify $\mathbf{x}$ in its correct class [23]. If the modeler has complete statistical knowledge of the underlying joint distribution of the observation $\mathbf{x}$ and the category θ_m $(m = 1,\ldots,M)$, a standard Bayes analysis will yield an optimal decision procedure and the corresponding minimum (Bayes) probability of error classification, R^*.

However, if the only knowledge the modeler has of the distribution is that which can be inferred from samples, then the decision to classify $\mathbf{x}$ into the category θ_m depends on the sample $\mathbf{x}_1,\ldots,\mathbf{x}_n$ along with its correct classification in categories $\theta_1,\ldots,\theta_M$, and the procedure is by no means clear. The classification problem falls into the domain of supervised classifications, where there is no an optimal classification procedure with regards to all underlying statistics.

Assuming that the classified samples $\mathbf{x}_i$ are independently identically distributed according to the distribution of $\mathbf{x}$, certain heuristic arguments may be considered about good decision procedures. For example, it is reasonable to assume that observations which are close together (in some appropriate distance metric) will have almost the same posterior probability distributions in their respective classifications.

Thus to classify the unknown sample $\mathbf{x}$ we may choose to give a heavier weight to the nearby $\mathbf{x}_i$'s. Perhaps the simplest non-parametric decision procedure of this type is the nearest neighbor (NN) classification method, which assigns the category of its nearest neighbor to $\mathbf{x}$.

[*] Corresponding author.
E-mail address: b.sierra@ehu.es (B. Sierra).
URL: http://www.sc.ehu.es/ccwrobot (B. Sierra).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.07.024

The first formulation of a rule of the NN type and primary previous contribution to the analysis of its properties is presumed to have been made by Fix and Hodges [13]. They investigated a method that is known as *k* Nearest Neighbors (*k*-NN), which assigns an unclassified point to the class most heavily represented among its *k* nearest neighbors.

In this paper we present a modification of the *k*-NN method that searches for the *k* nearest neighbors of the point to be classified in each class, and assigns the point to the class whose *k* points have the minimal mean distance to the new point. The idea is based on the assumption that the underlying distribution of the predictor variables (components of $\mathbf{x}$) could be different in each class.

This paper is organized as follows. In Section 2 we review the *k*-NN classification method, while Section 3 is devoted to Related Works in distance based classifiers, the new proposed method is introduced in Section 4, in Section 5 we show the experimental results obtained and concluding remarks are presented in Section 6.

2. The *k*-NN classification method

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a correctly classified sample in classes $\theta_1, \ldots, \theta_M$, where $\mathbf{x}_i$ takes values in a metric space upon which a distance function d is defined. We will consider the pairs $(\mathbf{x}_i, \theta^i)$, where $\mathbf{x}_i$ is the p-variate observation upon the ith individual, and θ^i is the class or category which that individual belongs to. We usually say that "$\mathbf{x}_i$ belongs to θ^i" when we mean precisely that the ith individual, upon which measurements $\mathbf{x}_i$ have been observed, belongs to category $\theta^i \in \{\theta_1, \ldots, \theta_M\}$.

Consider a new pair $(\mathbf{x}, \theta)$, where only the measurement $\mathbf{x}$ is observable, and where we estimate θ by using the information contained in the set of correctly classified points. We shall call

$$\mathbf{x}' \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$$

the *nearest neighbor* (NN) of $\mathbf{x}$ if

$$\min_{i=1,\ldots,n} d(\mathbf{x}_i, \mathbf{x}) = d(\mathbf{x}', \mathbf{x}).$$

The NN classification decision method gives to $\mathbf{x}$ the category θ^i, that is, the category of its nearest neighbor $\mathbf{x}_i$. In case of a tie between several neighbors, a modification of the decision rule is applied.

An immediate extension to this decision rule is the so called *k*-NN approach [7], which assigns the candidate $\mathbf{x}$ the class which is most frequently represented in the *k* nearest neighbors to $\mathbf{x}$. In Fig. 1, for example, the 3-NN decision rule would decide that class θ_o is active because two of the three nearest neighbors of $\mathbf{x}$ belong to class θ_o.

3. Related work

Much research has been devoted to the *k*-NN rule [8]. One of the most important results is that *k*-NN has a very good asymptotic performance. Broadly speaking, for a very large design set, the expected probability of incorrect classifications (error) R achievable with *k*-NN is bound as follows:

$$R^* < R < 2R^*$$

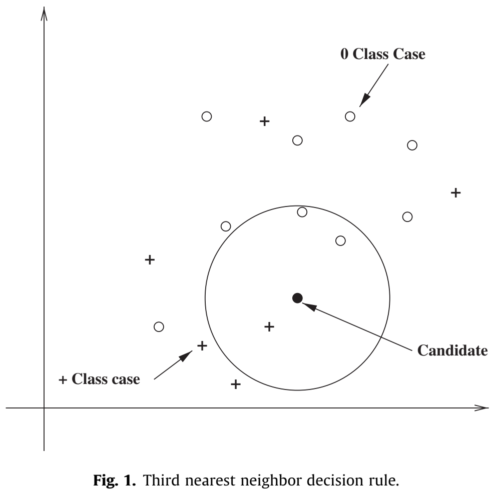


Fig. 1. Third nearest neighbor decision rule.

where R^* is the optimal (minimal) error rate for the underlying distributions. This performance, however, is demonstrated for the training set size tending to infinity, and thus, it is not really applicable to real world problems in which we usually have a training set of hundreds or thousands of cases, too few for the number of probability estimations to be performed.

Some distance based approaches, such as that of Weinberger et al. [34] try to increase the obtained accuracy in distance based classification by looking for a specific distance, in an automatic way, for each classification problem. The proposed approach could be used to deal with unbalanced or biased databases; a similar idea can be found in other distance based methods [32,18,14]. Alternatively, *PEBLS* instance based inducer (Cost and Salzberg [6]) incorporates MVDM distance metric to deal with symbolic features, a modification of Stanfill and Waltz's VDM metric [30].

Improvements in classification can also be achieved by selecting and/or weighting features (see [31] for an example). Probabilistic voting approaches have also been used [24,29]; the main idea here is that each case among the k nearest ones makes a weighted vote in favour of the class it belongs to, being the weight the probability each case has of belonging to its own class. A very well kown approach is the so called Instance Based Learning (IBL), based on the work of Aha [2] and Wettschereck [35]; there are several versions of the algorithm [1].

Another problem that arises with the distance based classifier systems is the management of large database. Studies devoted to data reduction [12] show interesting approaches which could also be used in combination with any distance based algorithm when the size of the database is huge; there are also studies that try to accelerate the execution of the distance based algorithm [33,21] to obtain faster classifications. Other approaches, such as Nearest neighbor editing, aim to increase the classifier generalization ability by removing noisy instances from the training set [15].

There are other distance based classifiers which aim to deal with the so called multi labeling problem [36], classifying each case in several categories. For instance, and taking as an example the document categorization area, a newspaper article reporting the wedding of the president of a certain country would obtain Politics and Society as category labels, both being adequate for the document.

The next section is devoted to the new approach we present in this paper.

4. The *k* Nearest Neighbor Equality method

The new approach can be seen as a new distance based classifier within the k-NN family of classifiers. Two are the main reasons for this new version: to decrease the influence of outliers in the voting process, and to include all the classes as candidates in the final selection process.

4.1. Motivation to extend k-NN

Whereas k-NN rule offers good results in general, there are some situations where it could be improved. Let us consider the following situation with 2 classes where the observations are drawn from bivariate normal populations:

$$\mathbf{X}_{\theta_1} \sim N(\mu_1, \Sigma) \quad \text{and} \quad \mathbf{X}_{\theta_2} \sim N(\mu_2 \Sigma),$$

with $\mu_1 = (3,0)', \mu_1 = (-3,0)'$ and variance–covariance matrix $\Sigma = \begin{pmatrix} 2^2 & 0 \\ 0 & 3^2 \end{pmatrix}$. We generated 40 cases from each class (see Fig. 2). Let us consider case $\mathbf{x} = (1, -1.5)$ which is more likely drawn from class θ_1 than from class θ_2. Under 3-NN, the closest cases are indicated as shaded symbols and $\mathbf{x}$ would be classified in class θ_2. However, we can see, in general, that cases from θ_1 (circles) are closer from $\mathbf{x}$ than cases from θ_2 (triangles). Summarizing, there are situations where a few cases from one class can "contaminate" the natural surroundings of the other. The method we propose addresses this problem and aims to solve it by diminishing the importance of outlier cases in the classification process.

We propose a modification of the k-NN algorithm that takes into account the information about the real class given by the values of the components of the observation $\mathbf{x}$.

In view of the interest in k-NN, it is surprising that – to the best of the authors knowledge – the following generalization of the rule has not been investigated: given k, search for the k nearest neighbor cases to $\mathbf{x}$ in each class $\theta_1, \ldots, \theta_M$, and classify the case $\mathbf{x}$ to the class θ_{m^*} whose k nearest neighbor cases have the *minimum mean* distance to $\mathbf{x}$. That is, considering individuals $\mathbf{x}_{i_m^1}, \ldots, \mathbf{x}_{i_m^k}$ are the k NN for $\mathbf{x}$ in class θ_m, $m = 1, \ldots, M$,

$$\text{classify } \mathbf{x} \text{ in } \theta_{m^*} \quad \text{if} \quad \min_{m=1,\ldots,M} \left\{ \frac{1}{k} \sum_{l=1,\ldots,k} d(\mathbf{x}, \mathbf{x}_{i_m^l}) \right\} = \frac{1}{k} \sum_{l} d(\mathbf{x}, \mathbf{x}_{i_{m*}^l}).$$

We have called this rule k Nearest Neighbor Equality (k-NNE).

This method will be introduced and empirically investigated below. It will be shown that its application outperforms k-NN, especially for multi-class problems ($M > 2$ classes).

The new proposed method, k-NNE is shown in its algorithmic form in Fig. 3. It is a simple method that obtains good results in multi-class problems. Although the computational cost seems to be expensive, it is very similar to the original k-NN computational cost, since in both cases the distance with respect to all cases must be calculated. Moreover, the obtained results are better than those obtained using other classification techniques (Classification Trees, Rule Induction, Instance Based Learning,...) found in the literature.

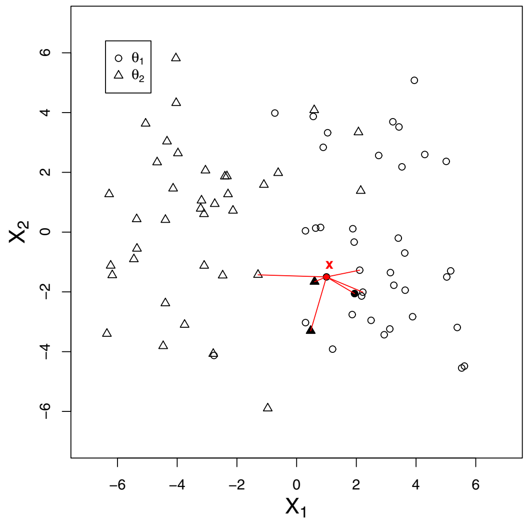


Fig. 2. Synthetic data to show situations where *k*-NNE can overcome some shortcomings of *k*-NN. The 3 closest cases are indicated as shaded symbols. The 3 closest cases in each class are indicated with connecting segments.

```
begin k-NNE
  As input we have the samples file, containing $n$ cases $(\mathbf{x}_i, \theta^i), i = 1, ..., n$,
  the value of $k$ and a new case $\mathbf{x}$ to be classified
  FOR each class value $\theta_m$ DO
    BEGIN
    Select the k nearest neighbors to $\mathbf{x}$ that belong to $\theta_m$ from the sample file
    Compute the mean distance from these $k$ points to $\mathbf{x}$, $\bar{d}_{\theta_m}$
    END
  Output the class $m^*$ with the minimal mean distance $\bar{d}_{\theta_{m^*}}$ among all the classes
end K-NNE
```

Fig. 3. The pseudo-code of the *k* Nearest Neighbor Equality Algorithm.

As can be seen in Fig. 4, the *k*-NNE approach works as follows: given a set of n correctly classified cases $(\mathbf{x}_1, \theta^1), (\mathbf{x}_2, \theta^2), \ldots, (\mathbf{x}_n, \theta^n)$ in a classification problem with M classes $\theta_1, \ldots, \theta_M$, given a new case to be classified $(\mathbf{x}, \theta)$ in which the class θ is unknown, and once a number k is fixed, the following classification process is performed:

For each class θ_m, search for the k nearest neighbors to $\mathbf{x}$ among the cases belonging to class θ_m: $\mathbf{x}_{i_m^1}, \ldots, \mathbf{x}_{i_m^k}$ $(m = 1, \ldots, M)$.

For the k nearest neighbor cases of $\mathbf{x}$ in each class θ_m, compute the mean distance $\bar{d}_{\theta_m}$:

$$\bar{d}_{\theta_m} = \frac{1}{k} \sum_{l=1,\ldots,k} d(\mathbf{x}, \mathbf{x}_{i_m^l}), \quad m = 1, \ldots, M.$$

Assign case $\mathbf{x}$ to the class θ_{m^*} which has the minimum mean distance $\bar{d}_{\theta_{m^*}}$, that is,

$$\hat{\theta} = \theta_{m^*} \text{ suchthat } \theta_{m^*} = \arg\ \min\{\bar{d}_{\theta_1}, \ldots, \bar{d}_{\theta_M}\}.$$

In principle, it would be possible to use another measure among the distances between the k NN in each class and case $\mathbf{x}$, for instance, the median. If the mean distance has been selected here, it is because it can identify scenarios with outliers. That is to say, when some outliers of a class θ_m are close to cases of any other class θ_l, $(l \neq m)$ and far from cases of the class they belong to (i.e., θ_m), $\bar{d}_{\theta_m}$ could become big, and therefore, the influence of the outlier in the voting process is diminished.

As in the *k*-NN classification technique, a tie-break method must be implemented. In this first approach to the method, we break ties by using the prior probabilities of the classes, that is, selecting the most probable class.

Returning to the situation shown in Fig. 2, the 3 NN to case $\mathbf{x}$ in each class θ_1 and θ_2 are indicated with connecting segments and the corresponding mean distances are $\bar{d}_{\theta_1} = 1.18$ and $\bar{d}_{\theta_2} = 1.27$. Hence, with 3-NNE case $\mathbf{x}$ will be classified in class θ_1 whereas by 3-NN it would be classified in class θ_2.

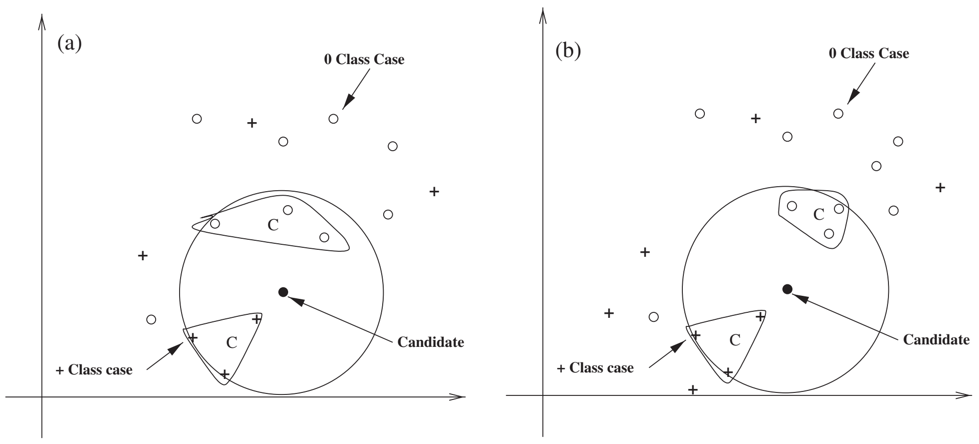


Fig. 4. 3-NNE Decision Rule compared with 6-NN. In Fig. 4a, for a first classification problem, both the 3-NNE and 6-NN result in a tie, while in Fig. 4b, for a second classification problem, the result of 3-NNE is the + class but the 6-NN has a tie.

Fig. 4 shows the behavior of the 3-NNE algorithm in comparison with the 6-NN for two two-class example problems. Although the computational cost is similar, the behavior of the 6-NNE method could make good use of the discriminant information provided by the predictor variables. This can be better seen in multi-class problems.

In Fig. 4 we can see the difference between the two methods applied, although the result given in this case example is the same (Fig. 4a), it can be seen that the methods follow different procedures in the classification process (Fig. 4b).

5. Experimental results

The characteristics of the experimental files are given in Table 1. These domains are public at the Statlog project WEB page [22], and we have searched for several multi-class problems to compare the behavior of our algorithm with other algorithms.

5.1. Classifiers

Supervised classifiers [10] from different families are chosen. Seven well known inducers are used in the experiments:

- *ID3* decision tree algorithm (Quinlan [25]). It does not prune the expanded tree.
- *C4.5* decision tree algorithm (Quinlan [26]). Instead of *ID3*, it makes a *post-pruning* phase, based on *error based pruning* algorithm.
- *Naive Bayes (NB)* algorithm (Cestnik [5]). It is based on Bayesian rules and, given that the value of the class is known, it assumes independence between the occurrences of feature values to predict the class.
- *Naive Bayes Tree (NBTree)* algorithm (Kohavi [19]). It runs *Naive Bayes* at the leaves of an induced decision tree. It normally improves *Naive Bayes* in large databases.

Table 1
Details of experimental domains.

Domain	Training cases	Test cases	Number of classes	Number of attributes
Glass	142	72	7	9
Iris	100	50	3	4
Letters	15,000	5000	26	16
Nettalk	7229	7242	324	203
Optdigit	3823	1797	10	64
Pendigit	7494	3493	10	16
Pima	200	332	2	7
Satimage	4435	2000	7	36
Shuttle	43,500	14,500	7	9
Vote	300	135	2	16

- *OneR* is a simple classifier that induces a set of rules, where each rule is based on only one predictor variable (the variable could be different in different rules), i.e., rules based on the value of a single attribute (Holte [16]).
- *CN2* rule induction classifier, based on the work of Clark and Nibblet [4]. It uses statistical tests to expand classification rules.
- *Neural Network* classifier based on the back-propagation algorithm[23]. Three layers have been used and a different number of intermediate neurons.

The class distribution of the training databases can be seen in Fig. 5. Different distributions appear, some of them are uniformly distributed, while in other cases the distribution is biased.

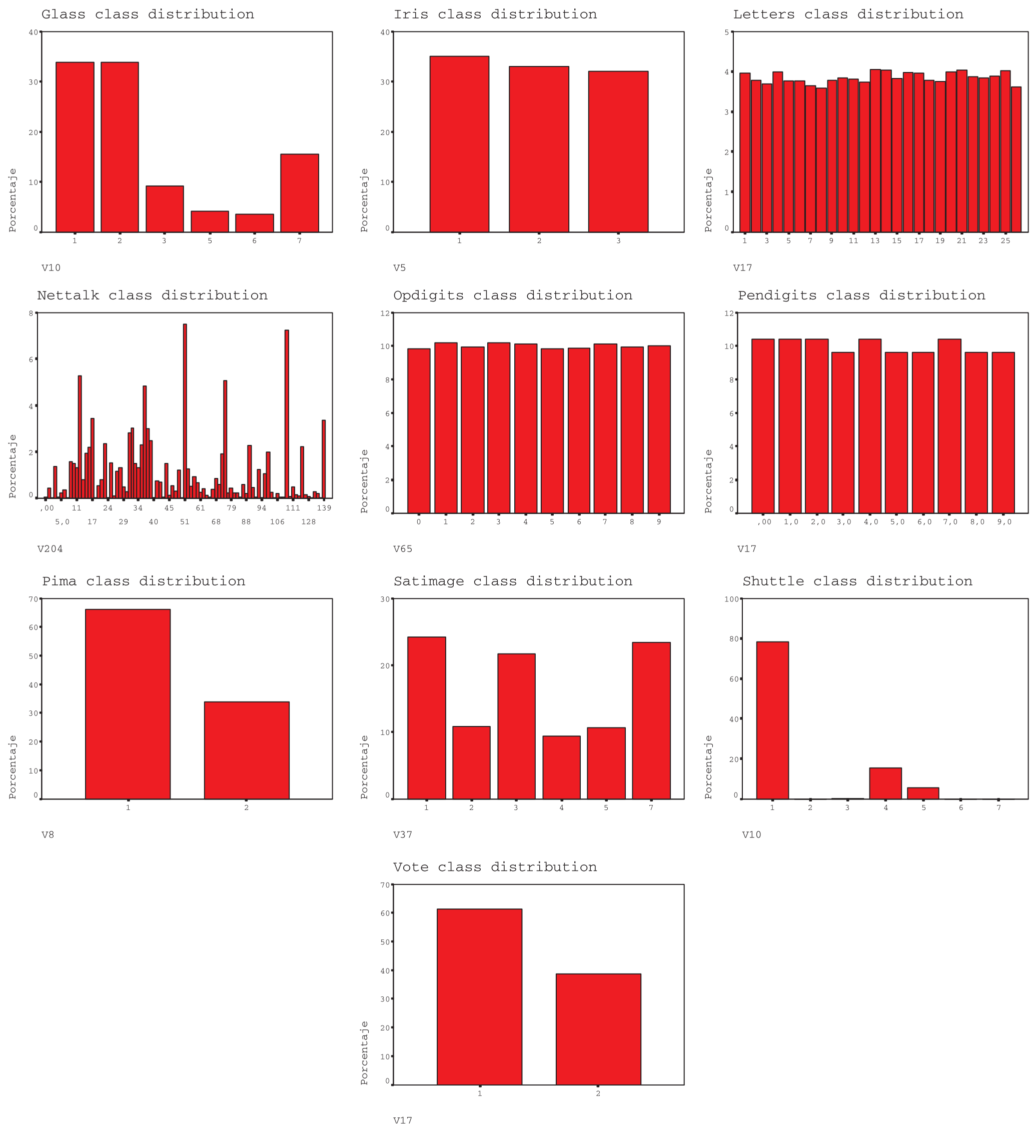


Fig. 5. Class distributions of the training databases.

5.2. Datasets

Ten databases are used to test our hypothesis. Most of them are obtained from the *UCI Machine Learning Repository* [3]. The *Nettalk* database was obtained from *MLC++* main repository [20]. All databases have a separate set of training data and testing data. The characteristics of the databases are given in Table 1.

We use the Training files as case examples for all the applied techniques, and the Test cases in order to estimate the error of the approach being used. Table 2 shows the experimental results obtained using standard Machine Learning approaches.

5.3. Distance based classifier comparison: training and test databases

In our experiments, we have run the *k*-NNE with different *k* values, and have compared the results obtained with those obtained with other methods implemented in standard Machine Learning software packages. In Table 3 we can see the results achieved with the *k*-NN method, while Table 4 shows the results obtained by the new proposed *k*-NNE method for some values of *k*. Obviously, the $k = 1$ case is equivalent to the NN method (or to the 1-NN method).

As can be seen in Tables 3 and 4, the new proposed method obtains a better accuracy than the *k*-NN method, especially in multi-class problems, i.e., in those problems in which the class number is high. For example, when applying the new method to the nettalk database (324 classes), the best accuracy result obtained is 66.53 in correctly classified percentage, while the best result obtained by the *k*-NN paradigm is 59.80. Although some other paradigms outperform this accuracy (ID3, 72.52, Neural Net 71.42), we are interested in how our proposed method compares with other distance based paradigms. When looking to the accuracy obtained with the letter database (26 classes), the *k*-NNE approach obtains a performance of

Table 2
Details of accuracy level percentages for the databases.

Inducer	Glass	Iris	Letters	Nettalk	Opdigit	Pendigit	Pima	Satimage	Shuttle	Vote
ID3	62.50	94.00	76.65	72.52	54.14	91.51	71.71	84.80	99.99	94.17
C4.5	62.50	92.00	87.02	71.50	56.93	92.02	75.39	85.40	99.95	97.04
NB	50.00	96.00	63.20	60.99	82.63	82.22	78.01	79.65	87.61	91.85
NBTree	63.89	96.00	84.30	65.85	89.93	92.74	78.91	81.65	99.98	95.53
oneR	48.61	94.00	16.52	12.48	23.20	36.60	73.82	58.80	94.67	97.04
CN2	76.40	94.00	64.00	75.40	67.40	86.40	75.80	71.30	99.40	95.60
Neural Net	63.89	98.00	89.24	71.42	97.11	93.25	80.42	82.20	98.87	97.04
Best result	**76.40**	**98.00**	**89.24**	**75.40**	**97.11**	**93.25**	**80.42**	**88.80**	**99.99**	**97.04**

Table 3
Accuracy level percentage of the *k*-NN method for the databases using different *K* numbers.

k	1	2	3	4	5	6	7	8	9	10
Glass	**81.94**	73.61	69.44	68.06	66.67	69.44	62.50	69.44	65.28	66.67
Iris	96.00	90.00	**98.00**	96.00	94.00	96.00	94.00	92.00	92.00	94.00
Letter	**95.62**	94.80	95.20	95.00	95.04	95.08	95.00	94.64	94.84	94.66
Nettalk	55.59	52.18	55.40	57.21	58.20	59.02	59.13	**59.67**	59.80	59.75
Opdigit	**98.00**	97.44	97.77	97.61	97.83	97.72	97.61	97.61	97.66	97.50
Pendigit	97.74	97.34	**97.77**	97.54	97.37	97.34	97.34	97.34	97.28	97.28
Pima	68.37	71.99	71.11	76.20	**78.92**	78.31	78.01	78.01	78.01	77.41
Satimage	89.45	88.95	**90.35**	90.25	90.35	89.80	89.85	89.25	89.40	89.50
Shuttle	**99.88**	99.81	99.83	99.81	99.80	99.78	99.79	99.77	99.79	99.75
Vote	**93.33**	92.59	91.85	92.59	93.33	93.33	93.33	91.85	92.59	92.59

Table 4
Accuracy level percentage of the *k*-NNE method for the seven databases using different *k* numbers.

k	1	2	3	4	5	6	7	8	9	10
Glass	**81.94**	79.17	70.83	69.44	68.06	66.67	63.89	63.89	63.89	62.50
Iris	96.00	96.00	**98.00**	98.00	96.00	96.00	96.00	92.00	92.00	92.00
Letter	95.62	96.08	**96.28**	96.28	96.26	95.96	95.80	95.76	95.42	95.16
Nettalk	55.59	60.66	64.24	65.30	66.21	**66.53**	66.43	66.16	65.66	65.53
Opdigit	98.00	98.33	**98.39**	98.22	98.11	98.00	97.89	97.83	97.77	97.66
Pendigit	97.74	98.00	**98.03**	97.97	97.83	97.74	97.74	97.54	97.54	97.48
Pima	68.37	74.10	**76.51**	76.51	75.60	75.90	76.20	76.20	75.90	76.20
Satimage	89.45	90.65	**90.80**	90.60	90.75	90.55	90.15	90.00	89.95	90.00
Shuttle	**99.88**	99.86	99.85	99.85	99.85	99.82	99.79	99.79	99.77	99.77
Vote	**93.33**	92.59	92.59	93.33	93.33	93.33	92.59	93.33	93.33	93.33

96.28%, while the *k*-NN best performance is 95.62%. For this database, both distance based paradigms outperform the results obtained with the other paradigms (as shown in Table 2).

This better performance of the presented approach with respect to the *k*-NN does not seem to hold when working with the databases whose class number is low (iris, vote); in both of them, the obtained performance is equivalent for the *k*-NN and *k*-NNE approaches.

In Table 5 the best results obtained for each database in each of the three previous experiments is presented in a summarized way.

We have applied classifier comparison statistical tests as recommended by Demsar [9], and no significance differences are obtained when either applying the Fisher-Snedecor test for the three rows or the Wilcoxon rank test for each pair.

Another value we can compare is the mean accuracy obtained for all the ten values of the *k* parameter. There are presented in Table 6. As can be seen, on average the *k*-NNE method outperforms the standard algorithm in 8 of the 10 databases.

This comparison is made because selecting the best *k* value for the test databases is not considered sound in the Machine Learning classifier selection process.

In order to better understand the results obtained, and following Demsar [9], a comparison among all *k* values is performed; in order to compare *k*-NN and *k*-NNE, we choose the 20 different results given by 20 different classifiers. As shown in Table 7, the best rank mean (4.65) is obtained by the *k*-NNE algorithm with a *k* value of 4. It can also be seen that the mean ranks of *k*-NNE are better in general, and in fact five of them are better than the best mean rank obtained by *k*-NN (9.60 for $k = 1$). The obtained Fisher–Snedecor $F_{9,9}$ value is 4.53, which does allow to consider *k*-NNE, with *k* equal to 4, as the best algorithm over all the others.

Table 5
Best percentages obtained for the databases in each of the three classifier subsets considered.

Classifier	Glass	Iris	Lett.	Nett.	Opd.	Pend.	Pima	Satim.	Shut.	Vote
Standard ML	76.40	**98.00**	89.24	**75.40**	97.11	93.25	**80.42**	88.80	**99.99**	**97.04**
k-NN	**81.94**	**98.00**	95.62	59.80	98.00	97.77	78.92	90.35	99.88	93.33
k-NNE	**81.94**	**98.00**	**96.28**	66.53	**98.39**	**98.03**	76.51	**90.80**	99.88	93.33

Table 6
Mean percentages of the *k*-NN and *k*-NNE methods for the ten values of *k* in each of the databases.

Classifier	Glass	Iris	Letters	Nettalk	Opdigit	Pendigit	Pima	Satimage	Shuttle	Vote
k-NN	69.30	94.2	94.99	57.60	97.68	97.43	75.63	89.71	99.80	92.74
k-NNE	69.03	95.2	95.86	64.23	98.02	97.76	75.15	90.29	99.82	93.10

Table 7
Mean percentages of the *k*-NN and *k*-NNE methods for the ten values of *k* in each of the databases.

Cl	*k*	Glass	Iris	Letters	Nettalk	Opdigit	Pendigit	Pima	Satimage	Shuttle	Vote	Mean
k-NN	1	1.50	7.50	8.50	17.50	6.0	7.50	19.50	16.50	1.50	6.0	9.20
	2	4.0	20.0	18.0	20.0	20.0	16.50	17.0	20.0	9.50	15.0	16.0
	3	7.50	2.0	11.0	19.0	11.50	5.0	18.0	6.50	7.0	19.50	10.70
	4	10.50	7.50	15.50	16.0	17.0	11.0	10.50	8.0	9.50	15.0	12.05
	5	13.0	13.0	14.0	15.0	9.50	14.0	1.0	6.50	11.0	6.0	10.30
	6	7.50	7.50	13.0	14.0	13.0	16.50	2.0	14.0	16.0	6.0	10.95
	7	19.50	13.0	15.50	13.0	17.0	16.50	4.0	13.0	13.50	6.0	13.10
	8	7.50	17.0	20.0	12.0	17.0	16.50	4.0	19.0	18.0	19.50	15.05
	9	15.0	17.0	17.0	10.0	14.50	19.50	4.0	18.0	13.50	15.0	14.35
	10	13.0	13.0	19.0	11.0	19.0	19.50	6.0	15.0	20.0	15.0	15.05
k-NNE	1	1.50	7.50	8.50	17.50	6.0	7.50	19.50	16.50	1.50	6.0	9.20
	2	3.0	7.50	4.0	9.0	2.0	2.0	16.0	3.0	3.0	15.0	6.45
	3	5.0	2.0	1.50	8.0	1.0	1.0	7.50	1.0	5.0	15.0	4.70
	4	7.50	2.0	1.50	7.0	3.0	3.0	7.50	4.0	5.0	6.0	**4.65**
	5	10.50	7.50	3.0	3.0	4.0	4.0	15.0	2.0	5.0	6.0	6.0
	6	13.0	7.50	5.0	1.0	6.0	7.50	13.50	5.0	8.0	6.0	7.25
	7	17.0	7.50	6.0	2.0	8.0	7.50	10.50	9.0	13.50	15.0	9.60
	8	17.0	17.0	7.0	4.0	9.50	11.0	10.50	10.50	13.50	6.0	10.60
	9	17.0	17.0	10.0	5.0	11.50	11.0	13.50	12.0	18.0	6.0	12.10
	10	19.50	17.0	12.0	6.0	14.50	13.0	10.50	10.50	18.0	6.0	12.70

5.4. Distance based classifier comparison: cross-validation approach

To make a better comparison between the *k*-NN algorithm and the new proposed *k*-NNE paradigm, we have carried out a more complete experiment dividing the data-files (which came as fixed training and test data files in the repository) into ten randomly generated training and test files, each of the same size shown in Table 1.

The results obtained by the two algorithms are shown in Fig. 6. As can be seen, the *k*-NN algorithm results are not as good as those obtained by the new proposed paradigm. The standard deviation is very low, and thus the statistical testing (using Wilcoxon range testing) indicates, in almost all the cases, that the new paradigm is more accurate than the standard *k*-NN.

The results obtained for the data in which the problems have more classes indicate that the approach presented here could work better when classifications among more than two classes have to be carried out:

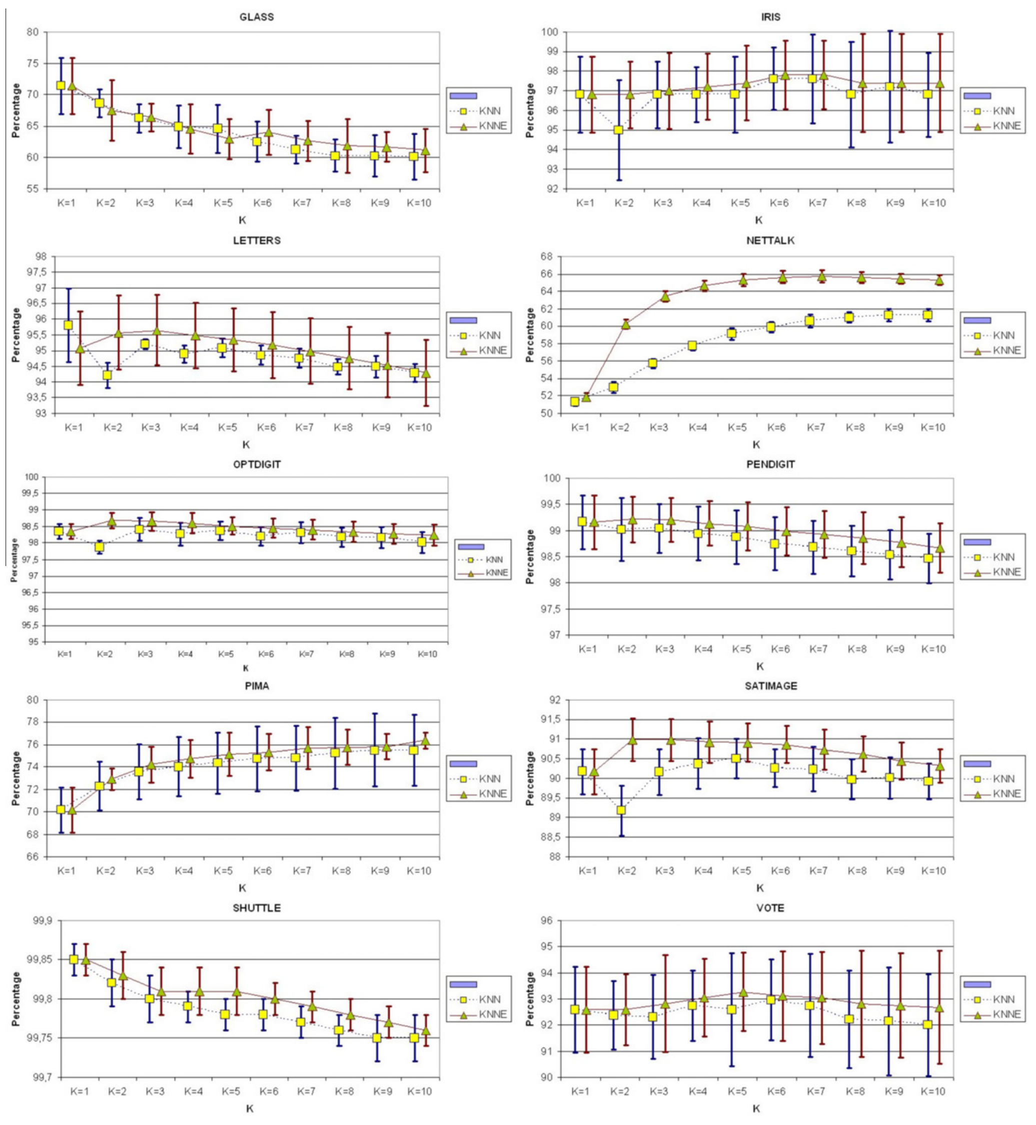


Fig. 6. Mean Accuracy level percentage and standard deviation obtained by the *k*-NN and *k*-NNE methods for the databases using different *k* numbers.

Table 8
Comparisons among the ten values of k for the k-NN and k-NNE methods. Best value obtained appears in bold.

Cl	*k*	Glass	Iris	Letters	Nettalk	Opdigit	Pendigit	Pima	Satimage	Shuttle	Vote	Mean
k-NN	1	1.50	15.50	8.0	19.50	9.50	3.50	19.50	15.50	1.50	13.50	10.75
	2	3.0	20.0	20.0	18.0	20.0	8.0	18.0	20.0	4.0	16.0	14.70
	3	6.0	15.50	5.0	17.0	6.0	7.0	16.0	8.0	8.50	17.0	10.60
	4	7.0	15.50	11.0	16.0	14.0	10.0	15.0	11.0	10.50	9.0	11.90
	5	8.50	15.50	8.0	15.0	8.0	12.0	13.0	9.0	13.0	13.50	11.55
	6	13.0	3.50	12.0	14.0	16.0	15.0	11.50	13.0	13.0	5.0	11.60
	7	16.0	3.50	14.0	12.0	12.0	16.0	10.0	14.0	15.50	9.0	12.20
	8	18.50	15.50	17.0	11.0	17.0	18.0	8.0	18.0	17.50	18.0	15.85
	9	18.50	9.50	16.0	9.0	18.0	19.0	5.0	17.0	19.50	19.0	15.05
	10	20.0	15.50	18.50	10.0	19.0	20.0	6.0	19.0	19.50	20.0	16.75
k-NNE	1	1.50	15.50	8.0	19.50	9.50	3.50	19.50	15.50	1.50	13.50	10.75
	2	4.0	15.50	2.0	13.0	1.0	1.0	17.0	1.50	3.0	13.50	7.15
	3	5.0	11.0	1.0	8.0	2.0	2.0	14.0	1.50	6.0	6.50	5.70
	4	8.50	9.50	3.0	7.0	3.0	5.0	11.50	3.0	6.0	3.50	6.0
	5	11.0	6.50	4.0	5.0	4.0	6.0	9.0	4.0	6.0	1.0	*5.65*
	6	10.0	1.50	6.0	2.0	5.0	9.0	7.0	5.0	8.50	2.0	**5.60**
	7	12.0	1.50	10.0	1.0	7.0	11.0	4.0	6.0	10.50	3.50	6.65
	8	14.0	6.50	13.0	3.0	11.0	13.0	3.0	7.0	13.0	6.50	9.0
	9	15.0	6.50	15.0	4.0	13.0	14.0	2.0	10.0	15.50	9.0	10.40
	10	17.0	6.50	18.50	6.0	15.0	17.0	1.0	12.0	17.50	11.0	12.15

- Letters: 26 classes, k-NN obtains 95.19 ± 00.16 mean accuracy and k-NNE 95.65 ± 01.12.
- Nettalk: 324 classes, k-NN obtains 61.24 ± 00.72 mean accuracy and k-NNE 65.72 ± 00.66.
- Optdigit: 10 classes,k-NN obtains 98.41 ± 00.34 mean accuracy and k-NNE 98.67 ± 00.23.
- Pendigit: 10 classes, k-NN obtains 99.16 ± 00.51 mean accuracy and k-NNE 99.21 ± 00.44.

In the experiment carried out, it can also be seen that the two algorithms offer a similar accuracy, but that the one presented in this paper outperforms the accuracy in all the multi-class problems.

Once again, as proposed by Demsar in [9], a comparison among all k values is performed. As shown in Table 8, the best rank mean (5.60) is obtained by the k-NNE algorithm with a k value of 6. It can also be seen that the mean ranks of k-NNE are better in general than the mean rank obtained by k-NN (10.60 for $k = 3$). Moreover, the obtained Fisher-Snedecor $F_{9,9}$ value is 4.92, which leads us to consider k-NNE with k equal to 6 as the best algorithm among the rest.

6. Conclusion and future work

A new method extending the k-NN idea is presented in this paper. The new method, called k-NNE, is based on the idea that predictor variables could have a different probability distribution in each class. Consequently, it searches for elements of all classes in the proximity of the new case to be classified.

This new method is used in different multi-class problems, and its final results are compared with those obtained by using the standard ML paradigms. We do not expect our new method to be better than that of k-NN one in all the classification problems, but it works better in almost all the experiments we have performed, and the difference is more evident when the problem has more than two classes.

Also we would like point out that the new method has been presented in its simplest distance calculation approach, and compared with the same version of the original k-NN. A lot of extensions could be applied to the algorithm: the distance metric, the weight of the neighbors depending on its distance, the weighing of the variables in the distance calculation, and so on. Different versions can be designed to try to decrease the number of calculations to be performed to obtain the k nearest neighbors of a given case [28,11]. Different techniques of prototype selection and/or attribute selection could also be used in some extensions of the new algorithm, as has been done with the k-NN algorithm [27,17,29]. The goal of these extensions would be to decrease the computation order of the algorithm in the distance calculation.

As further work we are going to apply different k-NN extensions to the k-NNE: weighting techniques, condensation methods, and editing approaches will be combined with the new proposed method in order to compare its behavior with that of the k-NN.

We are also collecting data from the Basque Country Weather Service in order to apply supervised classification techniques, including k-NNE, to the weather prediction task.

Acknowledgments

This work has been supported by the Basque Country University and by the Basque Government under the research team grant program.

References

[1] D. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Machine Learning 6 (1991) 37–66.

[2] D. Aha, Tolerating, irrelevant and novel attributes in instance-based learning algorithms, International Journal of Man–Machine Studies 36 (1) (1992) 267–287.

[3] B.L. Blake, C.J. Merz, UCI Repository of Machine Learning databases, Irvine, CA: University of California, Department of Information and Computer Science, 1998 (<http://www.ics.uci.edu/~mlearn/MLRepository.html>.

[4] P. Clark, T. Nibblet, The CN2 induction algorithm, Machine Learning 3 (4) (1989) 261–283.

[5] B. Cestnik, Estimating probabilities: a crucial task in machine learning, in: Proceedings of the European Conference on Artificial Intelligence, 1990 pp. 147–149.

[6] S. Cost, S. Salzberg, A weighted nearest neighbor algorithm for learning with symbolic features, Machine Learning 10 (1) (1993) 57–78.

[7] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Transactions IT-13 1 (1967) 21–27.

[8] B.V. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Recognition Classification Techniques, IEE Computer Society Press, 1991.

[9] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[10] T.G. Dietterich, Machine learning research: four current directions, AI Magazine 18 (4) (1997) 97–136.

[11] C. Domeniconi, D. Gunopulos, An Efficient approach for approximating multi-dimensional range queries and neares neighbor classification in large datasets, in: Proceedings of the Eighteenth International Conference on Machine Learning ICML01, 2001, pp. 98–105.

[12] H.A. Fayed, A.F. Atiya, A novel template reduction approach for the *K*-nearest neighbor method, IEEE Transactions on Neural Networks 20 (5) (2009) 890–896.

[13] E. Fix, J.L. Hodges Jr., Discriminatory Analysis, Nonparametric Discrimination, USAF school of Aviation Medicine, Randolf field, Project 21-49-004, Rept 4, 1951.

[14] Y. Gao, B. Zheng, G. Chen, Q. Li, C. Chen, G. Chen, Efficient mutual nearest neighbor query processing for moving object trajectories, Information Sciences 180 (11) (2010) 2176–2195.

[15] D. Guan, W. Yuan, Y.K. Lee, S. Lee, Nearest neighbor editing aided by unlabeled data, Information Sciences 179 (13) (2009) 2273–2282.

[16] R.C. Holte, Very simple classification rules perform well on most commonly used databases, Machine Learning 11 (1993) 63–90.

[17] I. Inza, P. Larrañaga, R. Etxeberria, B. Sierra, Feature subset selection by bayesian network-based optimization, Artificial Intelligence 123 (2000) 157–184.

[18] M.Z. Jahromi, E. Parvinnia, Robert John, A method of learning weighted similarity function to improve the performance of nearest neighbor, Information Sciences 179 (17) (2009) 2964–2973.

[19] R. Kohavi, Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996.

[20] R. Kohavi, D. Sommerfield, J. Dougherty, Data mining using MLC++, a machine learning library in C++, International Journal of Artificial Intelligence Tools 6 (4) (1997) 537–566. http://www.sgi.com/Technology/mlc/.

[21] Y.C. Liaw, M.L. Leou, C.M. Wu, Fast exact *K* nearest neighbors search using an orthogonal search tree, Pattern Recognition 43 (2010) 2351–2358.

[22] D. Michie, D.J. Spiegelhalter, C.C. Taylor (Eds.), Machine Learning, Neural and Statistical Classification (<http://www.amsta.leeds.ac.uk/~charles/statlog/>).

[23] T. Mitchell, Machine Learning, McGraw-Hill, 1997.

[24] J.M. Martínez-Otzeta, B. Sierra, Analysis of the iterated probabilistic weighted *k*-nearest neighbor method, a new distance-based algorithm, in: 6th International Conference on Enterprise Information Systems (ICEIS) 2, 2004, pp. 233–240.

[25] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.

[26] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman Publishers, Inc., Los Altos, California, 1993.

[27] A. Rozsypal, M. Kubat, Using the Genetic algorithm to reduce the size of a nearest-neighbor classifier and to select relevant attributes, in: Proceedings of the Eighteenth International Conference on Machine Learning ICML01, 2001, pp. 449–456.

[28] J. Schumacher, R. Bergmann, An effective approach for similarity-based retrieval on top of relational databases, 5th European Workshop on Case-Based Reasoning, Springer, 2000, pp. 273–284.

[29] B. Sierra, E. Lazkano, Probabilistic-weighted *K* nearest neighbor algorithm: a new approach for gene expression based classification, KES02 Proceedings, IOS press, 2002, pp. 932–939.

[30] C. Stanfill, D. Waltz, Toward memory-based reasoning, Communications of the ACM 29 (12) (1986) 1213–1228.

[31] M.A. Tahir, A. Bouridane, F. Kurugollu, Simultaneous feature selection and feature weighting using hybrid tabu search/*K*-nearest neighbor classifier, Pattern Recognition Letters archive 28 (2007) 438–446.

[32] S. Tan, Neighbor-weighted *K*-nearest neighbor for unbalanced text corpus, Expert Systems with Applications 28 (2005) 667–671.

[33] Q. Wang, S.R. Kulkarni, S. Verdu, Divergence estimation for multidimensional densities via *K*-nearest-neighbor distances, IEEE Transactions on Information Theory 55 (2009) 2392–2405.

[34] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, The Journal of Machine Learning Research 10 (2009) 207–244.

[35] D. Wettschereck, A Study of Distance-Based Machine Learning Algorithms, Ph.D. Thesis, Oregon State University, 1994.

[36] Z. Younes, F. Abdallah, T. Denux, Multi-label classification algorithm derived from *K*-nearest neighbor rule with label dependencies, in: 16th European Signal Processing Conference, 2008 (<http://www.eurasip.org/Proceedings/Eusipco/Eusipco2008/papers/1569101998.pdf>).

9

Surrounding Influenced K-Nearest Neighbors: A New Distance Based Classifier

Title: Surrounding Influenced K-Nearest Neighbors: A New Distance Based Classifier

Authors: I. Mendialdua, B. Sierra, E. Lazkano, I. Irigoien and E. Jauregi.

Conference: International Conference on Advanced Data Mining and Applications.

Publisher: Springer

DOI : http://dx.doi.org/10.1007/978-3-642-17316-5_26

Year: 2010

Surrounding Influenced K-Nearest Neighbors: A New Distance Based Classifier

I. Mendialdua, B. Sierra, E. Lazkano, I. Irigoien, and E. Jauregi

Department of Computer Science and Artificial Intelligence
University of the Basque Country
P. Manuel Lardizabal 1, 20018 Donostia-San Sebastian
Basque Country, Spain
{inigo.mendialdua,b.sierra,e.lazkano,itziar.irigoien,
ekaitz.jauregi}@ehu.es

Abstract. The nearest neighbor classification method assigns to an unclassified point the class of the nearest of a set of previously classified points. An extension to this approach is the *K*-NN method, in which the classification is made taking into account the *K* nearest points and classifying the unclassified point by a voting criteria from this k points. We present a new method that extends the *K*-NN limits, taking into account, for each neighbor, its *I* nearest neighbors. Experimental results are promising, obtaining better results for two class problems than the original *K*-NN.

Keywords: Nearest Neighbor, Supervised Classification.

1 Introduction

In supervised classification problems [1] there are two extremes of knowledge which the modeler may consider. Either (s)he may have complete statistical knowledge of the underlying joint distribution of the observation x and the category θ, or (s)he may have no knowledge of the underlying distribution except that which can be inferred from samples. In the first extreme, a standard Bayes analysis will yield an optimal decision procedure and the corresponding minimum (Bayes) probability of error classification R^*. In the other extreme, a decision to classify x into the category θ is allowed to depend only on a collection of n correct samples $(x_1, \theta_1), (x_2, \theta_2), ..., (x_n, \theta_n)$, and the decision procedure is by no means clear. This problem is in the domain of supervised classification, and no optimal classification procedure exists with respect to all underlying statistics.

If it is assumed that the classified samples (x_i, θ_i) are independently identically distributed according to the distribution of (x, θ), certain heuristic arguments may be made about good decision procedures. For example, it is reasonable to assume that observations which are close together (in some appropriate distance metric) will have almost the same posterior probability distributions on their respective classifications.

Thus to classify the unknown sample x we may wish to weight the evidence of the nearby x_i's most heavily. Perhaps the simplest non-parametric decision procedure of this form is the nearest neighbor (NN) classification method, which assigns to x the category of its nearest neighbor.

The first formulation of a rule of the NN type and primary previous contribution to the analysis of its properties it is presumed to have been made by Fix and Hodges [2]. They investigated a method that is known as K Nearest Neighbors (K-NN), which assigns to an unclassified point the class most heavily represented among its k nearest neighbors.

In this paper we present a modification to the K-NN method. Besides the K nearest neighbors, our method also considers their I surrounding neighbors classes. The objetive is to minimize the influence of the outliers in the final decission.

This paper is organized as follows. In section 2 we review the K-NN classification method while section 3 is devoted to Related Works in distance based classifiers; the new proposed method is introduced in section 4, in section 5 we show the experimental results obtained and in the final section 6 concluding remarks are presented..

2 The *K*-NN Classification Method

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a correctly classified sample in classes $\theta_1, \ldots, \theta_M$, where $\mathbf{x}_i$ takes values in a metric space upon which a distance function d is defined. We will consider the pairs $(\mathbf{x}_i, \theta^i)$ where $\mathbf{x}_i$ is the p-variate observation upon the ith individual and θ^i is the class or category which that individual belongs to. We usually say that "$\mathbf{x}_i$ belongs to θ^i" when we mean precisely that the ith individual, upon which measurements $\mathbf{x}_i$ have been observed, belongs to category $\theta^i \in \{\theta_1, \ldots, \theta_M\}$.

Consider a new pair $(\mathbf{x}, \theta)$, where only the measurement $\mathbf{x}$ is observable, and where we estimate θ by using the information contained in the set of correctly classified points. We shall call

$$\mathbf{x}' \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \ , \tag{1}$$

the *nearest neighbor* (NN) of $\mathbf{x}$ if

$$\min_{i=1,\ldots,n} d(\mathbf{x}_i, \mathbf{x}) = d(\mathbf{x}', \mathbf{x}) \ . \tag{2}$$

The NN classification decision method gives to $\mathbf{x}$ the category θ^i, precisely the category of its nearest neighbor $\mathbf{x}_i$. In case of tie between several neighbors, it has to be broken by modifying the decision rule.

An immediate extension to this decision rule is the so called K-NN approach [3], which assigns the candidate $\mathbf{x}$ the class which is most frequently represented in the k nearest neighbors to $\mathbf{x}$.

3 Related Work

Much research has been devoted to the K-NN rule [4]. One of the most important results is that K-NN has very good asymptotic performance. Broadly speaking, for a very large design set, the expected probability of incorrect classifications (error) R achievable with K-NN is bounded as follows:

$$R^* < R < 2R^* \ , \tag{3}$$

where R^* is the optimal (minimal) error rate for the underlying distributions. This performance, however, is demonstrated for the training set size tending to infinity, and thus, it is not really applicable to real world problems in which we usually have a training set of about hundreds or thousands cases, too few for the number of probability estimations to be performed.

Some distance based approaches, such that of Weinberger et al. [5] try to increase the obtained accuracy in distance based classification by looking for a specific distance, in an automatic way, for each classification problem. The proposed approach could be used to deal with unbalanced or biased databases; a similar idea can be found in other distance based methods [6]. By the other side, *PEBLS* instance based inducer (Cost and Salzberg [7]) incorporates MVDM distance metric to deal with symbolic features, a modification of Stanfill and Waltz's VDM metric [8].

Improvement in classification can also be obtained by selecting and/or weighting features (see [9] for an example). Probabilistic voting approaches have also been used ([10], [11]); the main idea here is that each case among the K nearest ones make a weighted vote in favour of the class it belongs to, being the weight the probability each case has to belong to its own class. A very well kown approach is the so called Instance Based Learning (IBL), based on the work of Aha [12] and Wettschereck [13]; there are several versions of the algorithm [14].

By the other side, there are distance based classifiers which aim to deal with the so called multi labeling problem [15], in which, given a new case to be classified, a different number of categories could be given to it. For instance, and taking as example the document categorization area, a newspaper article relating the wedding of some country president would obtain Politics and Society as category labels, being both adequate for the document.

4 Surrounding Influenced-*K*-Nearest Neighbors

When the modeler has to approach the K-NN classification problem, it depends in its K nearest neighbors and it is possible that some of this may be outliers, so in this case the assigned category is not suitable. In view of this problem we think in a new solution to solve this. This solution extends the limits to further data.

Our method is similar of K-NN method, Figure 1 shows that similarity. We get the K Nearest Neighbors of the data that we want to classify, but instead of select its category we look to the I nearest neighbors of each of the K points and

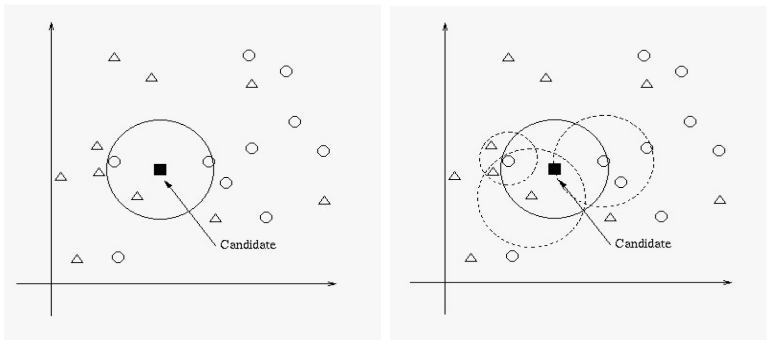


Fig. 1. Example of *SI*-*K*-NN algorithm where $K = 3$ and $I = 2$. First get K Nearest Neighbors and then for each K points get I Nearest Neighbors.

we take into account their categories. So we have two groups; the *K*-NN group, where there are the K nearest neighbors of the data that we are clasificating, and the *I*-NN group, where there are the I nearest neighbors of each K points. For all *I*x*K* neighbors we count the categories and the category which has the largest sum is assigned to the new data.

5 Experimental Results

5.1 Datasets

Twenty databases are used to test our hypothesis. All of them are obtained from the *UCI Machine Learning Repository* [16]. The characteristics of the databases are given in Table 1.

Table 1. Datasets used. Different characteristics of the twenty databases used in the experimental setup.

Domain	Num. of Instances	Num. of Attributes	Num. of Classes
Australian Credit	690	14	2
Balance	625	4	3
Blood	748	5	2
Breast Cancer	569	32	2
Car	1728	6	4
Glass	210	9	7
Haberman	306	3	2
Ionosphere	351	34	2
Iris	150	4	3
Letters	20000	16	26
Magic Telescope	19020	11	2
Optdigit	5620	64	10
Pendigit	10992	16	10
Pima	768	7	2
Spambase	4601	57	2
Statlog (Heart)	270	13	2
Statlog (Img Seg)	2310	19	7
Statlog (Landsat Sat)	6435	36	7
Statlog (Shuttle)	58000	9	7
Wine	178	13	3

5.2 Experiment-Setup

We have applied 5x2 fold cross-validation to each database [17]. In our experiments we have run the *SI*-*K*-NN with different *K* and *I* values, and we compare obtained results with the results that we obtained with *K*-NN method. In Table 2 we can see the results obtained with the *K*-NN method while Table 3 and Table 4 show the results obtained by the new proposed *SI*-*K*-NN method.

Table 2. Accuracy level percentage of the K-NN method for the databases using different K numbers

Datu-Basea	K = 1	K = 3	K = 5	K = 7	K = 9	Average
Australian Credit	80,927	84,231	86,028	85,681	86,202	84,614
Balance	77,820	81,474	83,974	87,179	87,756	83,641
Blood	69,679	74,010	74,171	75,454	75,294	73,721
Breast Cancer	95,633	96,760	96,901	96,830	96,901	96,605
Car	85,138	90,532	91,458	90,925	90,416	89,694
Glass	64,672	67,476	64,672	62,803	63,738	64,672
Haberman	64,705	66,405	68,366	70,457	70,849	68,156
Ionosphere	85,828	84,914	84,685	84,228	83,428	84,617
Iris	92,533	93,866	94,666	94,933	94,933	94,1864
Letters	94,338	94,276	93,966	93,598	93,126	93,8608
Magic Telescope	83,730	83,158	82,860	82,637	82,374	82,952
Opt Dig	98,398	98,476	98,370	98,163	98,085	98,298
Pen Dig	99,232	99,144	98,941	98,791	98,635	98,949
Pima	68,947	72,781	74,135	74,511	74,360	72,947
Spambase	88,852	89,060	89,139	88,939	88,426	88,883
Statlog (Heart)	76	79,851	81,333	81,481	81,481	80,029
Statlog (Img Seg)	95,844	95,168	94,233	94,129	93,991	94,673
Statlog (Landsat Sat)	89,574	90,071	89,686	89,344	89,070	89,549
Statlog (Shuttle)	99,930	99,866	99,822	99,782	99,739	99,828
Wine	94,382	95,730	96,404	96,179	96,629	95,865

In Table 5 we compared the results obtained with *K*-NN and *SI*-*K*-NN method. To make this comparison we have selected the average of *K*-NN and the best average between the *I* of *SI*-*K*-NN. As it can be seen, instead of sorting alphabetically, we have sort out the table depending on the number of classes that each database has. At first it may seem that the results are worse, but when it is sorted depending on the number of classes, it shows more interesting results. Viewed this way in multi-class problems, our method doesn't improve in any database. But in case that the class number is lower the results are better. When the number of classes is two the number of improved results is a bit more than the opposite, five better and four worse. It's the same when the number of classes is three, two better and one worse. The reason of this improvement is that our new method expands to a distant values, so the more classes there are, the more easier is to appear different classes and therefore the main class loses strength. Instead, the less classes there are, the less probability that the wrong class gains importance.

Table 3. Accuracy level percentage of the *SI-K*-NN method for the databases using different *K* and *I* numbers

		K = 1	K = 3	K = 5	K = 7	K = 9	Average
	I = 1	81,565	82,841	84,812	85,333	85,971	84,104
	I = 2	81,565	84,754	85,101	85,449	85,623	84,499
Australian Credit	I = 3	84,522	85,739	85,855	85,913	85,565	85,519
	I = 4	84,290	85,913	85,507	85,101	85,391	85,241
	I = 5	85,623	86,029	85,391	85,275	85,333	**85,530**
	I = 1	75,962	80,769	82,949	83,910	84,551	81,628
	I = 2	75,962	83,077	84,551	85,128	85,577	82,859
Balance	I = 3	76,218	85,000	85,705	85,705	85,897	**83,705**
	I = 4	75,833	84,872	85,385	85,385	86,090	83,513
	I = 5	77,051	84,103	85,000	85,000	85,385	83,308
	I = 1	70,214	70,588	71,230	70,749	71,872	70,930
	I = 2	73,155	72,834	73,583	74,118	74,866	73,711
Blood	I = 3	72,727	73,636	74,332	73,529	74,278	73,701
	I = 4	73,904	74,866	74,759	74,385	74,759	**74,535**
	I = 5	73,743	74,171	74,492	73,850	74,385	74,128
	I = 1	93,803	95,563	95,704	95,704	96,127	95,380
	I = 2	93,803	95,634	96,056	95,845	96,268	95,521
Breast	I = 3	93,873	95,845	96,127	95,915	96,197	95,592
	I = 4	94,577	95,845	95,915	95,915	95,845	95,620
	I = 5	95,775	95,986	96,127	95,704	95,986	**95,915**
	I = 1	76,204	81,505	82,778	83,241	84,144	81,574
	I = 2	82,755	83,958	84,699	85,486	85,810	**84,542**
Car	I = 3	79,097	83,657	84,838	85,139	85,926	83,731
	I = 4	81,713	83,912	84,884	85,255	85,486	84,250
	I = 5	80,347	83,796	84,699	85,116	85,394	83,870
	I = 1	59,626	61,121	60,374	60,000	60,000	60,224
	I = 2	59,626	62,243	61,121	60,187	60,561	**60,748**
Glass	I = 3	58,505	61,869	60,187	60,561	61,495	60,523
	I = 4	59,065	61,308	59,626	60,187	61,121	60,262
	I = 5	59,252	59,813	59,813	58,879	59,626	59,477
	I = 1	64,837	67,974	69,150	69,542	69,412	68,183
	I = 2	64,837	69,150	70,719	70,719	70,196	69,124
Haberman	I = 3	67,974	70,719	70,588	70,719	70,327	70,065
	I = 4	67,190	69,804	70,196	70,588	70,850	69,725
	I = 5	69,935	71,242	70,588	70,719	70,458	**70,588**
	I = 1	83,543	84,000	83,771	83,657	82,629	83,520
	I = 2	83,543	84,000	83,771	83,771	82,743	**83,566**
Ionosphere	I = 3	82,400	83,086	82,971	83,200	82,400	82,811
	I = 4	82,514	83,314	82,629	82,171	81,600	82,446
	I = 5	82,286	82,286	81,486	80,686	80,000	81,349
	I = 1	92,533	94,400	94,933	96,000	95,733	**94,720**
	I = 2	92,533	93,867	94,933	95,200	95,467	94,400
Iris	I = 3	92,267	94,667	95,200	95,733	95,467	94,667
	I = 4	92,533	94,400	94,933	95,733	95,200	94,560
	I = 5	92,267	94,667	94,933	95,733	95,467	94,613
	I = 1	90,622	91,572	91,750	91,506	91,156	**91,321**
	I = 2	90,622	91,654	91,408	91,066	90,536	91,057
Letters	I = 3	90,562	91,482	91,074	90,672	90,130	90,784
	I = 4	90,594	91,250	90,904	90,270	89,760	90,556
	I = 5	90,360	91,010	90,554	90,008	89,474	90,281
	I = 1	81,708	82,008	81,855	81,735	81,567	**81,775**
Magic Gamma	I = 2	81,708	81,924	81,720	81,577	81,468	81,679
Telescope	I = 3	81,708	81,886	81,685	81,499	81,367	81,629
	I = 4	81,966	81,836	81,655	81,457	81,340	81,651
	I = 5	81,794	81,718	81,493	81,335	81,184	81,505
	I = 1	97,900	98,221	98,100	97,922	97,915	**98,011**
	I = 2	97,900	98,157	98,078	97,865	97,829	97,966
Optical Digit	I = 3	97,765	98,121	98,007	97,843	97,815	97,910
	I = 4	97,786	98,121	98,000	97,786	97,737	97,886
	I = 5	97,715	98,093	97,957	97,786	97,722	97,855
	I = 1	98,897	98,967	98,759	98,661	98,537	**98,764**
	I = 2	98,897	98,952	98,766	98,624	98,472	98,742
Pen Dig	I = 3	98,810	98,876	98,675	98,574	98,428	98,672
	I = 4	98,857	98,846	98,643	98,490	98,395	98,646
	I = 5	98,712	98,715	98,530	98,428	98,341	98,545
	I = 1	72,105	72,256	71,729	73,083	73,158	72,466
	I = 2	72,105	74,511	74,586	74,286	73,835	**73,865**
Pima	I = 3	72,406	73,233	73,684	74,361	73,835	73,504
	I = 4	72,932	73,459	73,759	73,835	73,759	73,549
	I = 5	73,609	73,008	73,985	74,211	74,211	73,805

Table 4. Accuracy level percentage of the *SI-K*-NN method for the databases using different *K* and *I* numbers

		K = 1	K = 3	K = 5	K = 7	K = 9	Average
	I = 1	86,602	87,641	87,721	87,653	87,578	87,439
	I = 2	86,602	88,225	88,032	87,914	87,684	87,692
Sat Img	I = 3	87,087	88,119	87,846	87,833	87,504	87,678
	I = 4	87,293	88,287	87,858	87,709	87,529	**87,735**
	I = 5	87,386	87,883	87,684	87,566	87,324	87,569
	I = 1	99,869	99,857	99,815	99,782	99,734	**99,811**
	I = 2	99,876	99,827	99,819	99,779	99,732	99,806
Shuttle	I = 3	99,826	99,818	99,788	99,768	99,721	99,784
	I = 4	99,828	99,819	99,781	99,741	99,717	99,777
	I = 5	99,778	99,774	99,751	99,732	99,708	99,749
	I = 1	84,383	86,809	87,217	86,930	86,748	86,417
	I = 2	84,383	87,061	87,461	87,417	87,174	86,699
Spambase	I = 3	85,400	87,217	87,000	86,696	86,678	86,598
	I = 4	86,426	87,409	87,165	86,913	86,652	**86,913**
	I = 5	85,783	87,096	86,887	86,617	86,417	86,560
	I = 1	74,074	74,815	77,481	78,222	80,741	77,067
	I = 2	74,074	78,370	80,148	79,407	81,333	78,667
Statlog Heart	I = 3	78,074	80,741	80,741	81,333	82,222	80,622
	I = 4	79,556	80,889	80,741	81,778	82,815	81,156
	I = 5	81,481	80,889	81,185	82,074	82,370	**81,600**
	I = 1	93,108	93,870	93,576	93,455	93,299	**93,461**
Statlog Image	I = 2	93,108	93,680	93,662	93,558	93,091	93,420
Segmentation	I = 3	92,537	93,385	93,541	93,177	92,675	93,063
	I = 4	92,641	93,420	93,385	92,918	92,589	92,990
	I = 5	92,242	93,299	93,195	92,623	92,346	92,741
	I = 1	91,011	93,933	94,607	95,730	95,730	94,202
	I = 2	91,011	93,708	94,157	95,056	95,056	93,798
Wine	I = 3	91,685	93,933	95,056	95,730	95,281	94,337
	I = 4	92,360	95,506	95,506	95,730	95,955	**95,011**
	I = 5	91,910	95,056	95,281	95,730	96,404	94,876

Table 5. Best accuracy level percentage of the *K*-NN and *SI-K*-NN method for the databases sorting by the number of classe

Num. of Classes	Domain	K-NN	SI-K-NN
	Australian Credit	84,614	↑ 85,530
	Blood	73,721	↑ 74,535
	Breast Cancer	96,605	↓ 95,915
	Haberman	68,156	↑ 70,588
2 Classes	Ionosphere	84,617	↓ 83,566
	Magic Telescope	82,952	↓ 81,775
	Pima	72,94	↑ 73,865
	Spambase	88,883	↓ 86,913
	Statlog (Heart)	80,0296	↑ 81,600
	Balance	83,641	↑ 83,705
3 Classes	Iris	94,186	↑ 94,720
	Wine	95,865	↓ 95,011
	Car	89,694	↓ 84,542
	Glass	64,672	↓ 60,748
	Letters	93,860	↓ 91,321
>3 Classes	Opt Dig	98,298	↓ 98,011
	Pen Dig	98,949	↓ 98,764
	Statlog (Landsat Sat)	89,549	↓ 87,735
	Statlog (Shuttle)	99,828	↓ 99,811
	Statlog (Img Seg)	94,673	↓ 93,461

6 Conclusion and Further Results

A new method extending the *K*-NN idea is presented in this work. The new method, called *SI-K*-NN is created with the idea of reducing the influence of outliers in the final decision.

This new method is used in different problems, and its final results are compared with those obtained by using the K-NN method. We do not expect our new method to be better than the K-NN one in all the classification problems, but it works similar and in some cases improve the results.

As further work we are going to do new experiments in which we will consider the I and K nearest neighbors classes giving them different weights.

References

1. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
2. Fix, E., Hodges Jr, J.L.: Discriminatory analysis, nonparametric discrimination. Technical Report Project 21-49-004, USAF school of Aviation Medicine, Randolf field (1951)
3. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory 13, 21–27 (1967)
4. Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Recognition Classification Techniques. IEEE Computer Society Press, Los Alamitos (1991)
5. Weinberger, K.Q., Saul, L.K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. The Journal of Machine Learning Research 10, 207–244 (2009)
6. Tan, S.: Neighbor-Weighted K-Nearest Neighbor for Unbalanced Text Corpus. Expert Systems with Applications 28, 667–671 (2005)
7. Cost, S., Salzberg, S.: A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning 10, 57–78 (1993)
8. Stanfill, C., Waltz, D.: Toward Memory-Based Reasoning. Communications of the ACM 29(12), 1213–1228 (1986)
9. Tahir, M.A., Bouridane, A., Kurugollu, F.: Simultaneous Feature Selection and Feature Weighting Using Hybrid Tabu Search/K-Nearest Neighbor Classifier. Pattern Recognition Letters Archive 28, 438–446 (2007)
10. Martínez-Otzeta, J.M., Sierra, B.: Analysis of the Iterated Probabilistic Weighted k-Nearest Neighbor Method, a New Distance-Based Algorithm. 6th International Conference on Enterprise Information Systems (ICEIS), vol. 2, pp. 233–240. Porto (2004)
11. Sierra, B., Lazkano, E.: Probabilistic-Weighted k Nearest Neighbor Algorithm: a New Approach for Gene Expression Based Classification. In: Sierra, B., Lazkano, E. (eds.) Proceedings of KES 2002, pp. 932–939. IOS Press, Amsterdam (2002)
12. Aha, D.: Tolerating, Irrelevant and Novel Attributes in Instance-Based Learning Algorithms. International Journal of Man-Machine Studies 36, 267–287 (1992)
13. Wettschereck, D.: A Study of Distance-Based Machine Learning Algorithms. Ph.D. Thesis, Oregon State University (1994)
14. Aha, D., Kibler, D., Albert, M.K.: Instance-Based learning algorithms. Machine Learning 6, 37–66 (1991)
15. Younes, Z., Abdallah, F., Denux, T.: Multi-Label Classification Algorithm Derived from K-Nearest Neighbor Rule with Label Dependencies. In: Younes, Z., Abdallah, F. (eds.) 16th European Signal Processing Conference, Lausanne (2008) **Lausanne**
16. Blake, B.L., Merz, C.J.: UCI Repository of Machine Learning databases. Department of Information and Computer Science, University of California, Irvine (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
17. Demsar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)

10

Positive Predictive Value based dynamic K-Nearest Neighbor

Title: Positive Predictive Value based dynamic K-Nearest Neighbor

Authors: I. Mendialdua, N. Oses, B. Sierra and E. Lazkano.

Conference: International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Publisher: IOS Press

DOI : http://dx.doi.org/10.3233/978-1-61499-105-2-59

Year: 2012

Advances in Knowledge-Based and Intelligent Information and Engineering Systems 59
M. Graña et al. (Eds.)
IOS Press, 2012
© 2012 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-61499-105-2-59

Positive Predictive Value based dynamic *K*-Nearest Neighbor

Iñigo Mendialdua[1], Noelia Oses[2], Basilio Sierra[1] and Elena Lazkano[1]

[1] Department of Computer Science and Artificial Intelligence
University of Basque Country. http://www.sc.ehu.es/ccwrobot
[2] NOF Consulting. http://www.nofcon.com

Abstract. The K Nearest Neighbors classification method assigns to an unclassified observation the class which obtains the best results after a voting criteria is applied among the observation's K nearest, previously classified points. In a validation process the optimal K is selected for each database and all the cases are classified with this K value. However the optimal K for the database does not have to be the optimal K for all the points. In view of that, we propose a new version where the K value is selected dynamically. The new unclassified case is classified with different K values. And looking for each K how many votes has obtained the winning class, we select the class of the most reliable one. To calculate the reliability, we use the Positive Predictive Value (PPV) that we obtain after a validation process. The new algorithm is tested on several datasets and it is compared with the K-Nearest Neighbor rule.

Keywords: Nearest Neighbor, Supervised Classification, Machine Learning, Non-parametric Pattern Recognition

1 Introduction

In supervised classification problems [1] there are two extremes of knowledge which the modeler may consider. Either (s)he may have complete statistical knowledge of the underlying joint distribution of the observation x and the category θ, or (s)he may have no knowledge of the underlying distribution except that which can be inferred from samples. In the first extreme, a standard Bayes analysis will yield an optimal decision procedure and the corresponding minimum (Bayes) probability of error classification R^*. In the other extreme, a decision to classify x into the category θ is allowed to depend only on a collection of n correct samples $(x_1, \theta_1), (x_2, \theta_2), ..., (x_n, \theta_n)$, and the decision procedure is by no means clear. This problem is in the domain of supervised classification, and no optimal classification procedure exists with respect to all underlying statistics.

If it is assumed that the classified samples (x_i, θ_i) are independently identically distributed according to the distribution of (x, θ), certain heuristic arguments may be made about good decision procedures. For example, it is reasonable to assume that observations which are close together (in some appropriate distance metric) will have almost the same posterior probability distributions on their respective classifications.

Thus to classify the unknown sample x we may wish to weigh the evidence of the nearby x_i's most heavily. Perhaps the simplest non-parametric decision procedure of this form is the nearest neighbor (NN) classification method, which assigns to x the category of its nearest neighbor.

The first formulation of a rule of the NN type and primary previous contribution to the analysis of its properties is presumed to have been made by Fix and Hodges [2]. They investigated a method that is known as K Nearest Neighbors (K-NN), which assigns to an unclassified point the class most heavily represented among its k nearest neighbors.

In this paper we present a new version of K-NN method which finds the most reliable K for each new case. In a validation process we calculate the success rate when a class is predicted for different K values and for different amounts that belongs to the predicted class between these K Nearest Neighbors. Thus, when a new case has to be classified, we will select the K which get the most reliable result.

This paper is organized as follows. In section 2 we review the K-NN classification method while section 3 is devoted to related work in distance-based classifiers. The new proposed method is introduced in section 5. Section 6 shows the experimental results obtained and in the final section concluding remarks are presented.

2 The *K*-NN Classification Method

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a correctly classified sample in classes $\theta_1, \ldots, \theta_M$, where $\mathbf{x}_i$ takes values in a metric space upon which a distance function d is defined. We will consider the pairs $(\mathbf{x}_i, \theta^i)$ where $\mathbf{x}_i$ is the p-variate observation upon the ith individual and θ^i is the class or category which that individual belongs to. We usually say that "$\mathbf{x}_i$ belongs to θ^i" when we mean precisely that the ith individual, upon which measurements $\mathbf{x}_i$ have been observed, belongs to category $\theta^i \in \{\theta_1, \ldots, \theta_M\}$.

Consider a new pair $(\mathbf{x}, \theta)$, where only the measurement $\mathbf{x}$ is observable, and where we estimate θ by using the information contained in the set of correctly classified points. We shall call $\mathbf{x}'$ the *nearest neighbor* (NN) of $\mathbf{x}$ if

$$\min_{i=1,\ldots,n} d(\mathbf{x}_i, \mathbf{x}) = d(\mathbf{x}', \mathbf{x}) \,. \tag{1}$$

The NN classification decision method gives to $\mathbf{x}$ the category θ^i, precisely the category of its nearest neighbor $\mathbf{x}_i$. In case of a tie between several neighbors, it has to be broken by modifying the decision rule.

An immediate extension to this decision rule is the so called K-NN approach [3], which assigns the candidate $\mathbf{x}$ the class which is most frequently represented among the k nearest neighbors of $\mathbf{x}$. In Figure 1, for example, the 3-NN decision rule would decide the class θ_o is active because two of the three nearest neighbors of $\mathbf{x}$ belong to class θ_o.

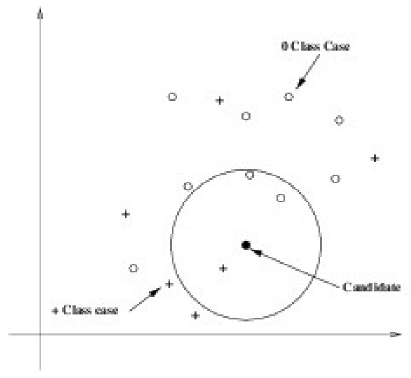


Fig. 1. Third Nearest Neighbor Decision Rule

3 Related work

Much research has been devoted to the K-NN rule [4]. One of the most important results is that K-NN has very good asymptotic performance. Broadly speaking, for a very large design set, the expected probability of incorrect classifications (error) R achievable with K-NN is bounded as follows:

$$R^* < R < 2R^* \tag{2}$$

where R^* is the optimal (minimal) error rate for the underlying distributions. This performance, however, is demonstrated for the training set size tending to infinity, and thus, it is not really applicable to real world problems in which we usually have a training set of hundreds or thousands cases, too few for the number of probability estimations to be performed.

Some distance based approaches, such that of Weinberger et al. [5] try to increase the obtained accuracy in distance based classification by looking for a specific distance, in an automatic way, for each classification problem. The proposed approach could be used to deal with unbalanced or biased databases; a similar idea can be found in other distance based methods [6]. On the other hand, *PEBLS* instance based inducer (Cost and Salzberg [7]) incorporates MVDM distance metric to deal with symbolic features, a modification of Stanfill and Waltz's VDM metric [8].

Improvement in classification can also be obtained by selecting and/or weighting features (see [9] for an example). Probabilistic voting approaches have also been used ([10], [11]); the main idea here is that each case among the K nearest ones make a weighted vote in favor of the class it belongs to, being the weight the probability each case has to belong to its own class. A very well known approach is the so called Instance Based Learning (IBL), based on the work of Aha [12] and Wettschereck [13]; there are several versions of the algorithm [14].

Another problem that arises with distance-based classifier systems is the management of large databases. Works devoted to data reduction [15] show interesting approaches which could also be used in combination with any other distance based algorithm when the size of the database is significant; other works in this field try to accelerate the execution of the distance based algorithm ([16], [17]) to obtain faster classifications.

Additionally, there are distance based classifiers which aim to deal with the so called multi labeling problem [18], in which, given a new case to be classified, a different number of categories could be given to it. For instance, and taking as example the document categorization area, a newspaper article relating the wedding of some country president would obtain Politics and Society as category labels, being both adequate for the document.

4 PPV *K*-NN

The standard procedure to use the K-NN algorithm with a new database corresponding to a classification problem, is to first select the appropriate K value for the datafile (K=5, for instance) and fix this value to deal with all the cases that have to be classified. The way the most suitable K value is obtained is mainly based on a validation process, although there are other possibilities found in the literature, for example Wang et al. [19] proposed a new method that dynamically adjust the number of nearest neighbors based on the statistical confidence.

As in Wang et al. work, in this paper we propose a new approach to determine the most likely value of K for each of the new cases to be classified. To do that, a validation approach is used as well, not to select a fixed K parameter for all the cases, but to select which K values are best for each of the different categories of the classification problem for the instance being processed.

To calculate which is the best K value for each new case, we use the Positive Predictive Value (PPV). The PPV is the proportion of instances which predicted to belong to certain class and are correctly classified. The Positive Predictive Value is defined as

$$PPV = \frac{numberofTruePositive}{numberofTruePositive + numberofFalsePositive} \tag{3}$$

where a True Positive is the event that the test makes a positive prediction, and the subject has a positive result. And a False Positive is the event that the test makes a positive prediction, and the subject has a negative result.

In a validation process, before the proper classifier is used, we calculate the PPV values for each predicted class, C_{Pr}, with different K values. As a result, for each C_{Pr} we obtain a PPV table which contains the different PPV values for each K value. Table 1 shows an example for a 3 class problem and K values from 1 to 4. We want to emphasize that the C_{Pr} is the class that a classifier predicted for the instance and no the class that the instance belongs to.

In order to make a deeper analysis of the classifier behavior and based on the results obtained for high values of K, we realize that it is not the same confidence

Table 1. Example of the PPV Tables, first version

C_{Pr}=1		C_{Pr}=2		C_{Pr}=3	
K	PPV	K	PPV	K	PPV
1	PPV_1	1	PPV_1	1	PPV_1
2	PPV_2	2	PPV_2	2	PPV_2
3	PPV_3	3	PPV_3	3	PPV_3
4	PPV_4	4	PPV_4	4	PPV_4

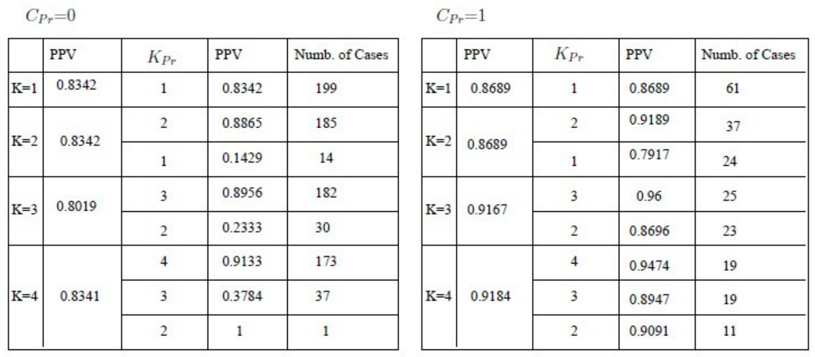
that 8 of the 9 neighbors belong to the C_{Pr} or to belong 5. This is the reason why we have extended the tables shown in Table 1 and take in consideration not only the K value and the C_{Pr}; we also take in consideration the amount of the cases that belongs to the C_{Pr} between the K neighbors, K_{Pr}. Hence, the tables that are used in classification task are extended to those shown in Table 2.

Table 2. Example of the PPV Tables, improved version

C_{Pr}=1			C_{Pr}=2			C_{Pr}=3		
K	K_{Pr}	PPV	K	K_{Pr}	PPV	K	K_{Pr}	PPV
1	1	PPV_1	1	1	PPV_1	1	1	PPV_1
2	2	PPV_2^2	2	2	PPV_2^2	2	2	PPV_2^2
	1	PPV_2^1		1	PPV_2^1		1	PPV_2^1
3	3	PPV_3^3	3	3	PPV_3^3	3	3	PPV_3^3
	2	PPV_3^2		2	PPV_3^2		2	PPV_3^2
	1	PPV_3^1		1	PPV_3^1		1	PPV_3^1
4	4	PPV_4^4	4	4	PPV_4^4	4	4	PPV_4^4
	3	PPV_4^3		3	PPV_4^3		3	PPV_4^3
	2	PPV_4^2		2	PPV_4^2		2	PPV_4^2
	1	PPV_4^1		1	PPV_4^1		1	PPV_4^1

Thus when a new case to be classified arrives, firstly we process it for different K values. For each K value, we get the C_{Pr} and the K_{Pr} and from the PPV tables we obtain the PPV value that correspond for each K. After that, we select the K with the highest PPV value and finally the K-NN result of this concrete K is assigned to the new case.

In Figure 2 we could see with an example how our algorithm works. In Figure 2(a) we could see the PPV tables that we obtain after a validation process. These tables show the real values that we got in our experiments with Ionosphere database. As it is a two-class problem there are two PPV tables, one for each C_{Pr}. On the other hand, in Figure 2(b) we could see the steps that our algorithm follows to classify a new case. In the first step, we get the 4 nearest neighbors classes. After that, in the second step, for different K values we apply the K-NN method obtaining the C_{Pr} and the K_{Pr}. In the third step, from the PPV tables we obtain the PPV value which belongs to the K, C_{Pr} and K_{Pr} values. And finally we select the case with the highest PPV value, and we assign to the new case its C_{Pr}.



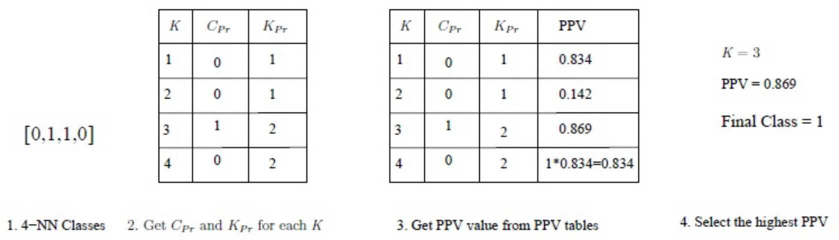
C_{Pr}=0

	PPV	K_{Pr}	PPV	Numb. of Cases
K=1	0.8342	1	0.8342	199
K=2	0.8342	2	0.8865	185
		1	0.1429	14
K=3	0.8019	3	0.8956	182
		2	0.2333	30
K=4	0.8341	4	0.9133	173
		3	0.3784	37
		2	1	1

C_{Pr}=1

	PPV	K_{Pr}	PPV	Numb. of Cases
K=1	0.8689	1	0.8689	61
K=2	0.8689	2	0.9189	37
		1	0.7917	24
K=3	0.9167	3	0.96	25
		2	0.8696	23
K=4	0.9184	4	0.9474	19
		3	0.8947	19
		2	0.9091	11

(a) PPV table for each C_{Pr}



[0,1,1,0]

K	C_{Pr}	K_{Pr}
1	0	1
2	0	1
3	1	2
4	0	2

K	C_{Pr}	K_{Pr}	PPV
1	0	1	0.834
2	0	1	0.142
3	1	2	0.869
4	0	2	1*0.834=0.834

$K = 3$
PPV = 0.869
Final Class = 1

1. 4−NN Classes
2. Get C_{Pr} and K_{Pr} for each K
3. Get PPV value from PPV tables
4. Select the highest PPV

(b) Step by step

Fig. 2. In 2(a) we show the tables that our algorithm get in our experiments after a validation process with Ionosphere database. Each C_{Pr} table shows the PPV for different K, with different K_{Pr}. In 2(b) we show the steps that our algorithm follows to assign the final class

Sometimes it is possible to be few cases to achieve the PPV value. For example, in Figure 2(a), in $C_{Pr} = 0$ Table, we could see that when $K = 4$ and $K_{Pr} = 2$, there is only one case. As this case has been classified correctly in the validation process, its success rate is very high. This success rate is not very reliable because they do not have enough cases to support this claim, and it is possible that the case correctly classified in the validation process could be an exception. To avoid these cases we have included a threshold. If there is no more than 10 cases, the degree of confidence is multiplied by the degree of confidence of the K. We could see that in Figure 2(b) for $K = 4$ case.

5 Experimental Results

In this section we show the experimental results obtained with different databases. We have compared our method with K-NN for different K values.

5.1 Datasets

Twenty-six databases are used to test our hypothesis. All of them are obtained from the *UCI Machine Learning Repository* [20]. The characteristics of the databases are given in Table 3.

Table 3. The characteristics of the 26 databases used in this experiment

Domain	Num. of Instances	Num. of Attributes	Num. of Classes
Australian Credit	690	14	2
Balance	625	4	3
Blood Transfusion	748	5	2
Breast Cancer	569	32	2
Car	1728	6	4
Cmc	1473	9	3
Diabetes	768	8	2
Glass	210	9	7
Haberman	306	3	2
Image Segmentation	2310	19	7
Ionosphere	351	34	2
Iris	150	4	3
Letters	20000	16	26
Magic Gamma Telescope	19020	11	2
Optical Digits	5620	64	10
Pen-Based Digits	10992	16	10
Sonar	208	60	2
Spambase	4601	57	2
Statlog(German)	1000	20	2
Statlog (Heart)	270	13	2
Statlog (Landsat)	6435	36	7
Statlog (Shuttle)	58000	9	7
Tic Tac Toe	958	9	2
Vowel Context	528	10	11
Waveform	5000	21	3
Wine	178	13	3

5.2 Experimental setup

In order to give a real perspective we have applied 5x2 fold cross validation to each database [21]. But firstly our algorithm needs a validation process to get the PPV values. So we have applied 5-hold out for each fold, where we have used the %70 as training and %30 as testing.

In Table 4 we show the results obtained by the K-NN method for different K values, where the best result is written in boldface. We do not include the K=2 value in this Table, because in the case of draw we give the preference to the nearest neighbor, and hence the K=2 and K=1 results always are the same.

In Table 5 we compare our method with the K-NN. Looking at the results of Table 5, it could be seen that in most of the cases our method improves

Table 4. Accuracy level percentage of the K-NN method

Database	K-NN=1	K-NN=3	K-NN=4	K-NN=5	K-NN=6	K-NN=7	K-NN=8	K-NN=9	Avg	σ
Australian Credit	80.203	83.275	83.565	84.812	84.783	84.870	84.957	85.072	83.942	1.6561
Balance	78.814	81.571	83.109	84.808	86.571	87.436	87.660	87.917	84.736	3.3180
Blood Transfusion	68.797	73.396	73.743	75.374	75.802	76.337	76.417	77.406	74.659	2.7275
Breast Cancer	95.458	96.725	96.761	96.444	96.725	96.444	96.585	96.232	96.422	0.4296
Car	85.509	90.660	90.880	91.551	91.123	90.949	90.856	90.729	90.282	1.9483
Cmc	43.505	45.516	46.250	46.685	46.766	47.242	46.834	46.861	46.208	1.2078
Diabetes	69.474	74.323	73.308	75.602	74.023	75.489	75.451	75.789	74.182	2.1008
Glass	64.299	65.421	63.271	64.860	63.084	63.645	62.523	63.178	63.785	0.9878
Haberman	65.098	68.039	67.908	69.281	69.412	71.830	71.699	73.203	69.559	2.6199
Image Segmentation	95.792	94.909	94.814	94.035	94.424	93.844	94.147	93.706	94.459	0.6897
Ionosphere	85.486	84.171	84.400	83.771	84.171	83.600	83.600	82.743	83.993	0.7909
Iris	93.867	94.400	94.400	94.933	94.267	94.267	94.267	94.933	94.417	0.3595
Letters	94.343	94.291	94.422	94.016	94.098	93.647	93.609	93.217	93.955	0.4251
Magic Gamma Telescope	80.124	82.211	82.462	82.980	83.115	83.212	83.281	83.213	82.575	1.0636
Optical Digits	98.359	98.456	98.491	98.349	98.406	98.185	98.256	98.139	98.330	0.1263
Pen-Based Digits	99.212	99.128	99.148	98.970	99.034	98.810	98.866	98.675	98.981	0.1859
Sonar	82.788	77.692	79.423	74.519	75.385	70.192	72.019	68.173	75.024	4.8764
Spambase	88.530	88.965	89.583	88.757	89.435	88.596	89.178	88.130	88.897	0.4893
Statlog (German)	66.960	69.200	68.840	70.000	70.520	70.620	71.580	71.000	69.840	1.4723
Statlog (Landsat)	89.473	90.002	89.983	89.890	90.045	89.557	89.765	89.442	89.770	0.2478
Statlog (Heart)	75.778	78.963	78.889	81.037	80.222	80.370	80.519	80.593	79.546	1.7060
Statlog (Shuttle)	99.932	99.876	99.880	99.835	99.835	99.794	99.792	99.751	99.837	0.0581
Tic-tac-toe	100.000	99.311	99.478	97.745	98.455	96.013	97.641	95.177	97.978	1.6995
VowelContext	93.758	80.202	76.687	63.737	59.758	49.394	44.626	38.646	63.351	19.1055
Waveform	77.160	80.448	80.348	81.748	81.476	82.696	82.504	83.044	81.178	1.9020
Wine	94.719	95.506	95.843	95.506	95.730	96.180	96.292	96.742	95.815	0.6117

the mean of all the K, only getting worse results in 5 databases. This can be considered logical because in Table 4, it can be seen that in some databases there are considerable differences for different values of K, which makes to devalue the mean.

On the other hand, if we compare our method with the best K, we could see that it improves in 6 databases, it draws in 1 and get worse in 20. Although this results do not seem promising, looking in more detail, we could see that the majority of the improved databases have low standard deviation value, σ, for the different K values. There are 12 databases which has $\sigma < 1$, and the results are very similar; in 5 of them our method gets the best result, in 1 they draw and in the other 6 the best K-NN is the optimal. We think that these are optimistic results taking into account that we are comparing our methods only result with the best between the K-NN's 9 results.

6 Conclusion and Further Results

A new method extending the K-NN idea is presented in this work: PPV K-NN. The main reason of this approach is to ensure that the final decision is made with some confidence. For doing that we search the most reliable K for each new case.

The new method has been implemented and tested over 26 databases from the UCI repository. We have compared our method with the best result and the mean of K-NN. Getting interesting results for the databases with regular results for different K in K-NN method.

Table 5. Comparisson between our method resultd with K-NN's best result and mean

Database	PPV K-NN	K-NN Best	Avg K-NN
Australian Credit	84.116	**85.072**	83.942
Balance	86.635	**87.917**	84.736
Blood Transfusion	76.364	**77.406**	74.659
Breast Cancer	96.655	**96.761**	96.422
Car	90.197	**91.551**	90.282
Cmc	**47.541**	47.242	46.208
Diabetes	73.947	**75.789**	74.182
Glass	64.766	**65.421**	63.785
Haberman	72.484	**73.203**	69.559
Image Segmentation	95.446	**95.792**	94.459
Ionosphere	**88.286**	85.486	83.993
Iris	94.133	**94.933**	94.417
Letters	**94.659**	94.422	93.955
Magic Gamma Telescope	82.387	**83.281**	82.575
Optical Digits	**98.491**	**98.491**	98.330
Pen-Based Digits	**99.216**	99.212	98.981
Sonar	79.904	**82.788**	75.024
Spambase	89.561	**89.583**	88.897
Statlog (German)	70.700	**71.580**	69.840
Statlog (Heart)	79.778	**81.037**	79.546
Statlog (Landsat)	**90.228**	90.045	89.770
Statlog (Shuttle)	99.924	**99.932**	99.837
Tic-tac-toe	99.937	**100.000**	97.978
VowelContext	92.667	**93.758**	63.351
Waveform	80.720	**83.044**	81.178
Wine	**96.966**	96.742	95.815

As future work it would be interesting to include the reject option. Where we reject the cases where there are two different K with a high PPV value that predict different classes.

References

1. Mitchell T.: Machine Learning. McGraw-Hill (1997)
2. Fix, E., Hodges Jr, J.L.: Discriminatory analysis, nonparametric discrimination. Technical Report Project 21-49-004, USAF school of Aviation Medicine, Randolf field (1951)
3. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification. IEEE Trans. IT-13, vol. 1, pp. 21-27 (1967)
4. Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Recognition Classification Techniques. IEEE Computer Society Press (1991)
5. Weinberger, K.Q., Saul, L.K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. The Journal of Machine Learning Research. vol. 10, pp. 207-244 (2009)
6. Tan, S.: Neighbor-Weighted K-Nearest Neighbor for Unbalanced Text Corpus. Expert Systems with Applications. vol. 28, pp. 667-671 (2005)
7. Cost, S., Salzberg, S.: A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning, vol. 10, num. 1 , pp. 57-78 (1993)
8. Stanfill, C., Waltz, D.: Toward Memory-Based Reasoning. Communications of the ACM, vol. 29, num. 12, pp. 1213-1228 (1986)
9. Tahir, M.A., Bouridane, A., Kurugollu, F.: Simultaneous Feature Selection and Feature Weighting Using Hybrid Tabu Search/K-Nearest Neighbor Classifier. Pattern Recognition Letters archive, vol. 28, pp. 438-446 (2007)

10. Martnez-Otzeta, J.M., Sierra, B.: Analysis of the Iterated Probabilistic Weighted k-Nearest Neighbor Method, a New Distance-Based Algorithm. 6th International Conference on Enterprise Information Systems (ICEIS), vol. 2, pp. 233-240 (2004)
11. Sierra, B., and Lazkano, E.: Probabilistic-Weighted k Nearest Neighbor Algorithm: a New Approach for Gene Expression Based Classification. KES02 proceedings (IOS press), pp. 932-939 (2002)
12. Aha, D.: Tolerating, Irrelevant and Novel Attributes in Instance-Based Learning Algorithms. International Journal of Man-Machine Studies, vol. 36, pp. 267-287 (1992)
13. Wettschereck, D.: A Study of Distance-Based Machine Learning Algorithms. Ph.D. Thesis, Oregon State University (1994)
14. Aha, D., Kibler, D., Albert, M.K.: Instance-Based learning algorithms. Machine Learning, vol. 6, pp. 37-66 (1991)
15. Fayed, H.A., Atiya, A.F.: A Novel Template Reduction Approach for the K-Nearest Neighbor Method. IEEE Transactions on Neural Networks, vol. 20, num. 5, pp. 890-896 (2009)
16. Wang, Q., Kulkarni, S.R., Verdu, S.: Divergence Estimation for Multidimensional Densities Via K-Nearest-Neighbor Distances. IEEE Transactions on Information Theory, vol. 55, pp. 2392-2405 (2009)
17. Liaw, Y.C., Leou, M.L., Wu, C.M.: Fast Exact K Nearest Neighbors Search Using an Orthogonal Search Tree. Pattern Recognition, vol. 43, pp. 2351-2358 (2010)
18. Younes, Z., Abdallah, F., Denux, T.: Multi-Label Classification Algorithm Derived from K-Nearest Neighbor Rule with Label Dependencies. 16th European Signal Processing Conference (2008) , C., Boto, F., Soraluze, I., Perez, A.: An incremental and hierarchical k-NN classifier for handwritten characters. In: Proc. 16th Internat. Conf. on Pattern Recognition, Quebec City, Canada, vol. 3, pp. 98-101 (2002)
19. Wang, J., Neskovic, P., Cooper, L.N.: Neighborhood selection in the k-nearest neighbor rule using statistical confidence. Pattern Recognition, vol. 39 pp. 417423 (2006)
20. Frank, A., Asuncion, A.: UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, Irvine, CA: University of California, School of Information and Computer Science (2010)
21. Demsar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, vol. 7, pp. 1-30 (2006)

11

Decreasing K Nearest Neighbor

Title: Decreasing K Nearest Neighbor

Authors: I. Mendialdua, B. Sierra, E. Lazkano, I. Irigoien and E. Jauregi.

Conference: Conferencia de la Asociación Española para la Inteligencia Artificial

Year: 2011

Decreasing K Nearest Neighbor

I. Mendialdua, B. Sierra, E. Lazkano, I. Irigoien and E. Jauregi

Department of Computer Science and Artificial Intelligence
University of the Basque Country
P. Manuel Lardizabal 1, 20018 Donostia-San Sebastián
Basque Country, Spain
{inigo.mendialdua, b.sierra, e.lazkano, itziar.irigoien, ekaitz.jauregi}@ehu.es

Resumen The K Nearest Neighbor classification method assigns to an unclassified point the class which obtains best results after a voting criterion is applied among its *K* previously classified nearest points. There are different aspects which have to be taken into account in order to use this paradigm: the parameter K to be used, the voting method to apply, the appropriate distance, among others. In this paper we present a new version *DK*-NN where we seek *K* value in which the most voted class must exceed a percentage of votes to be assigned. If the most voted class do not exceed this threshold the value of *K* will decrease until it is found a K which meet the condition. In this way, the experimenter does not have to select a fixed number of *K* before using the classifier. We have made several experiments with different threshold values and we have compared these with the K-NN method. Obtained results support the adequateness of the new distance based classification paradigm.

Keywords: Nearest Neighbor, Supervised Classification, Machine Learning, Nonparametric Pattern Recognition

1. Introduction

In supervised classification problems [1] there are two extremes of knowledge which the modeler may consider. Either (s)he may have complete statistical knowledge of the underlying joint distribution of the observation x and the category θ, or (s)he may have no knowledge of the underlying distribution except that which can be inferred from samples. In the first extreme, a standard Bayes analysis will yield an optimal decision procedure and the corresponding minimum (Bayes) probability of error classification R^*. In the other extreme, a decision to classify x into the category θ is allowed to depend only on a collection of n correct samples $(x_1,\theta_1),(x_2,\theta_2),...,(x_n,\theta_n)$, and the decision procedure is by no means clear. This problem is in the domain of supervised classification, and no optimal classification procedure exists with respect to all underlying statistics.

If it is assumed that the classified samples (x_i,θ_i) are independently identically distributed according to the distribution of (x,θ), certain heuristic arguments may be made about good decision procedures. For example, it is reasonable to assume that observations which are close together (in some appropriate distance metric) will have almost the same posterior probability distributions on their respective classifications.

Thus to classify the unknown sample x we may wish to weight the evidence of the nearby x_i's most heavily. Perhaps the simplest non-parametric decision procedure of this form is the nearest neighbor (NN) classification method, which assigns to x the category of its nearest neighbor.

The first formulation of a rule of the NN type and primary previous contribution to the analysis of its properties it is presumed to have been made by Fix and Hodges [2]. They investigated a method that is known as K Nearest Neighbors (K-NN), which assigns to an unclassified point the class most heavily represented among its k nearest neighbors.

One problem of this algorithm is that it assigns the final decision to the most voted class, regardless of its likelihood. In this paper we present a different way to use K-NN method, where if for one K the classifier has not a minimum of certainty to assign a class, the K value is decrease until it finds a K that meets the minimum uncertainty.

This paper is organized as follows. In section 2 we review the K-NN classification method while section 3 is devoted to Related Works in distance based classifiers; the new proposed method is introduced in section 4, in section 5 we show the experimental results obtained and in the final section 6 concluding remarks are presented..

2. The *K*-NN Classification Method

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be a correctly classified sample in classes $\theta_1, \ldots, \theta_M$, where $\mathbf{x}_i$ takes values in a metric space upon which a distance function d is defined. We will consider the pairs $(\mathbf{x}_i, \theta^i)$ where $\mathbf{x}_i$ is the p-variate observation upon the ith individual and θ^i is the class or category which that individual belongs to.

We usually say that "$\mathbf{x}_i$ belongs to θ^i"when we mean precisely that the ith individual, upon which measurements $\mathbf{x}_i$ have been observed, belongs to category $\theta^i \in \{\theta_1, \ldots, \theta_M\}$.

Consider a new pair $(\mathbf{x}, \theta)$, where only the measurement $\mathbf{x}$ is observable, and where we estimate θ by using the information contained in the set of correctly classified points. We shall call

$$\mathbf{x}' \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \tag{1}$$

the *nearest neighbor* (NN) of $\mathbf{x}$ if

$$\underset{i=1,\ldots,n}{\text{mín}} \, d(\mathbf{x}_i, \mathbf{x}) = d(\mathbf{x}', \mathbf{x}) \,, \tag{2}$$

The NN classification decision method gives to $\mathbf{x}$ the category θ^i, precisely the category of its nearest neighbor $\mathbf{x}_i$. In case of tie between several neighbors, it has to be broken by modifying the decision rule.

An immediate extension to this decision rule is the so called K-NN approach [3], which assigns the candidate $\mathbf{x}$ the class which is most frequently represented in the k nearest neighbors to $\mathbf{x}$.

3. Related work

Much research has been devoted to the K-NN rule [4]. One of the most important results is that K-NN has very good asymptotic performance. Broadly speaking, for a very large design set, the expected probability of incorrect classifications (error) *R* achievable with *K*-NN is bounded as follows:

$$R^* < R < 2R^* \tag{3}$$

where R^* is the optimal (minimal) error rate for the underlying distributions. This performance, however, is demonstrated for the training set size tending to infinity, and thus, it is not really applicable to real world problems in which we usually have a training set of about hundreds or thousands cases, too few for the number of probability estimations to be performed.

Some distance based approaches, such that of Weinberger et al. [5] try to increase the obtained accuracy in distance based classification by looking for a specific distance, in an automatic way, for each classification problem. The proposed approach could be used to deal with unbalanced or biased databases; a similar idea can be found in other distance based methods [6]. By the other side, *PEBLS* instance based inducer (Cost and Salzberg [7]) incorporates MVDM distance metric to deal with symbolic features, a modification of Stanfill and Waltz's VDM metric [8].

Improvement in classification can also be obtained by selecting and/or weighting features (see [9] for an example). Probabilistic voting approaches have also been used ([10], [11]); the main idea here is that each case among the *K* nearest ones make a weighted vote in favour of the class it belongs to, being the weight the probability each case has to belong to its own class. A very well known approach is the so called Instance Based Learning (IBL), based on the work of Aha [12] and Wettschereck [13]; there are several versions of the algorithm [14].

Other problem that arises with the distance based classifier systems is the large database management. Works devoted to data reduction [15] show interesting approaches which could also be used in combination with any other distance based algorithm when the size of the database is huge; there are as well works that try to accelerate the execution of the distance based algorithm ([16], [17]) to obtain faster classifications.

By the other side, there are distance based classifiers developed to deal with the so called multi labeling problem [18], in which, given a new case to be classified, a different number of categories could be given to it. For instance, and taking as example the document categorization area, a newspaper article relating the wedding of some country president would obtain

Next section is devoted to the new approach we present in this paper.

4. Decreasing *K* Nearest Neighbor

One weaknesses of the *K*-NN algorithm is that sometimes it makes a decision without enough certainty about the answer it is giving, mainly due to weak majority results. Trying somehow to avoid this, in our work we present a new method in which we select *K* value which gives high level of certainty in the classification process.

In 1970 Hellman [19] proposed the (*K*,*L*)-Nearest Neighbor rule, where it rejected patterns with higher risk of being misclassified. He introduced a new threshold (*L*) that indicated which was the minimum amount of votes that a class had to receive to be selected. If there is no class which got more votes than the indicated threshold, this case is not classified. Rodriguez et al. [20] created a hierarchy for handwritting characters using (*K*,*L*)-Nearest Neighbor algorithm. Their main objective was to reduce the computational cost. In the first level of the hierarchy they used a small train database for (*K*,*L*)-Nearest Neighbor; in this way the cases that were easy to classify are classified quickly. In the second one they used a bigger training database to classify more difficult cases that were rejected in the first level. And in the last one they use the Weighted *K*-NN to classify the most difficult cases.

Our method is an extension of Hellman algorithm taking as a reference Rodriguez et al. idea. In Hellman approach if any class exceeded the threshold the new case is not classified. Instead of that, while the new case is rejected, we propose to decrease sequentially the number of the *K* value until $K = 1$, in this case we apply the NN approach. The threshold is updated maintaining the same percentage.

In Figure 1 it could be seen an example of *DK*-NN. In Figure 1(a) where $K = 9$, the threshold $L = 7$ and there is no class which gets this amount of votes so this case is rejected. In Figure 1(b) we could see next step where $K = 5$ and $L = 4$, in this case neither the threshold is exceed. And in Figure 1(c) we could see the last step where it is assigned the Nearest Neighbor to the new case.

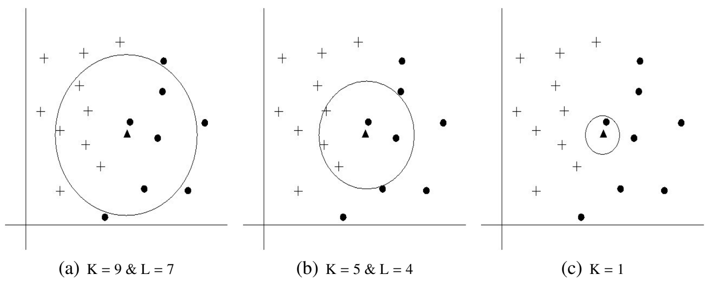


Figura 1. Example of *DK*-NN algorithm

5. Experimental Results

In this section we show the experimental results obtained with different databases. We have compared *DK*-NN with *K*-NN for different *K* values. To make these experiments we have used a software for machine learning called Weka (Waikato Environment for Knowledge Analysis) [21].

5.1. Datasets

Twelve databases are used to test our hypothesis. All of them are obtained from the *UCI Machine Learning Repository* [22]. The characteristics of the databases are given in Table 5.1.

Cuadro 1. The characteristics of the 12 databases used in this experiment

Domain	Instances	Attributes	Classes
Annealing	798	38	5
Cmc	1493	9	3
Dermatology	366	33	6
Glass	210	9	7
Iris	150	4	3
Nursery	12960	8	5
Optdigit	5620	64	10
Pendigit	10992	16	10
Statlog (Img Seg)	2310	19	7
Statlog (Landsat Sat)	6435	36	7
Statlog (Vehicle)	946	18	4
Wine	178	13	3

5.2. Experimental setup

In order to give a real perspective we have applied 5x2 fold cross validation to each database [23]. In our experiments we have ran *DK*-NN with different thresholds and we compare obtained results with the results obtained with *K*-NN.

In Table 5.2 we show the results obtained by the *K*-NN method for different *K* values, where the best result is written in boldface. In Table 5.2, 5.2 and 5.2 we show the results obtained by *DK*-NN for different reject thresholds. Furthermore, through these Tables it can be seen how our algorithm acts for different databases, showing for each step the percentage of classified cases and what percentage were correctly classified.

Cuadro 2. Accuracy level percentage of the *K*-NN method using different *K* numbers

K	1	2	3	4	5	6	7	8	9	Avg
Annealing	**0.93497**	0.93051	0.91670	0.91002	0.89599	0.89488	0.87906	0.87906	0.86993	0.90124
Cmc	0.42376	0.44277	0.45227	0.46083	0.45621	0.46857	0.46327	0.46897	**0.46993**	0.45629
Derm.	0.95082	0.95082	**0.96339**	**0.96339**	0.95683	0.96284	0.95902	0.96011	0.95519	0.95804
Glass	**0.67477**	0.65327	0.63925	0.64299	0.63084	0.62430	0.61308	0.61682	0.62430	0.63551
ImgSeg	**0.95550**	0.94511	0.94658	0.94199	0.94009	0.93870	0.93645	0.93602	0.93558	0.94178
Iris	0.94267	0.94800	**0.95867**	0.95333	0.95733	0.95733	0.95600	0.95600	0.95200	0.95348
Landsat	0.89461	0.89013	**0.89896**	0.89570	0.89806	0.89514	0.89411	0.89259	0.89308	0.89471
Nursery	0.88463	0.89846	0.92573	0.92824	0.93684	0.93818	0.94182	0.94219	**0.94377**	0.92665
Optdig	0.98370	0.97918	**0.98391**	0.98228	0.98242	0.98149	0.98125	0.98064	0.98085	0.98175
Pendig	**0.99212**	0.99034	0.99103	0.98997	0.98937	0.98865	0.98788	0.98741	0.98637	0.98924
Vehicle	0.66927	0.66998	0.67281	0.67352	**0.67541**	0.66809	0.66076	0.65981	0.66407	0.66819
Wine	0.95281	0.93146	**0.95618**	0.94382	0.95506	0.94494	0.95506	0.94944	0.95281	0.94906

Looking at the results, it could be seen that when threshold is 0.8, *DK*-NN gets better results than the best *K* for the *K*-NN algorithm in 5 of the 12 databases, while the other thresholds improve in the same 4 databases. Watching more closely, one can see that the results are interesting in the databases where among the used *K* values, there are not remarkable differences in the obtained results, i.e. when the standard deviation is low. This standard deviation can be seen in Table 5.2. There are 8 databases which has a regular results; our method gets better results in 4 of them, and even it gets worse in the other 4, it could be seen that the best threshold is very close to the best *K*. On the

Cuadro 3. Accuracy level percentage when threshold is 0.8

Thresh.=0.8	K=9 L=7		K=5 L=4		K=1		
Database	Perc. Cases	Well C.	Perc. Cases	Well C.	Perc. Cases	Well C.	Accuracy
Annealing	0.84699	0.93873	0.05991	0.81784	0.09310	0.77990	0.91670
Cmc	0.12003	0.65837	0.16388	0.50041	0.71609	0.38149	0.43422
Derm.	0.87049	0.98933	0.05956	0.84404	0.06995	0.68750	0.95956
Glass	0.38505	0.84466	0.14019	0.61333	0.47477	0.58071	↑ 0.68692
Image	0.86996	0.98358	0.05316	0.81922	0.07688	0.70270	0.95325
Iris	0.89200	0.98655	0.05733	0.79070	0.05067	0.39474	0.94533
Landsat	0.83969	0.95507	0.05047	0.71367	0.10984	0.56989	↑ 0.90057
Nursery	0.61201	0.99889	0.08789	0.97313	0.30011	0.66391	0.89610
Optdig	0.95712	0.99368	0.01897	0.90056	0.02391	0.70982	↑ 0.98512
Pendig	0.97853	0.99470	0.00970	0.89869	0.01177	0.80371	0.99152
Vehicle	0.34917	0.90657	0.17139	0.68828	0.47943	0.50740	↑ 0.67778
Wine	0.90449	0.98012	0.03483	0.87097	0.06067	0.66667	↑ 0.95730

Cuadro 4. Accuracy level percentage when threshold is 0.7

Thresh.=0.7	K=8 L=6		K=4 L=3		K=1		
Database	Perc. Cases	Well C.	Perc. Cases	Well C.	Perc. Cases	Well C.	Accuracy
Annealing	0.87016	0.93115	0.08263	0.79515	0.04722	0.77358	0.91247
Cmc	0.18303	0.61499	0.29532	0.46023	0.52166	0.36413	0.43842
Derm.	0.89891	0.98602	0.06831	0.74400	0.03279	0.70000	0.96011
Glass	0.45794	0.81020	0.25234	0.54444	0.28972	0.56774	0.67290
Image	0.89290	0.98032	0.06130	0.73729	0.04580	0.67297	0.95134
Iris	0.92667	0.97986	0.04667	0.74286	0.02667	0.35000	0.95200
Landsat	0.86474	0.94781	0.06847	0.67817	0.06679	0.52815	↑ 0.90132
Nursery	0.68850	0.99713	0.11006	0.91139	0.20144	0.59864	0.90742
Optdig	0.96648	0.99275	0.02036	0.83566	0.01317	0.64054	↑ 0.98491
Pendig	0.98333	0.99395	0.01072	0.86927	0.00595	0.74618	0.99114
Vehicle	0.41844	0.87966	0.25910	0.61040	0.32246	0.46848	↑ 0.67730
Wine	0.92022	0.97558	0.04831	0.81395	0.03146	0.67857	↑ 0.95843

Cuadro 5. Accuracy level percentage when threshold is 0.6

Thresh. = 0.6	K=9 L=6		K=6 L=4		K=3 L=2		K =1		
Database	Perc. Cases	Well C.	Perc. Cases	Well C.	Perc. Cases	Well C.	Perc. Cases	Well C.	Accuracy
Annealing	0.91537	0.90487	0.04811	0.73148	0.03385	0.73684	0.00267	0.91667	0.89087
Cmc	0.30102	0.57330	0.20937	0.43969	0.34121	0.40191	0.14840	0.34309	0.45268
Derm.	0.93388	0.97601	0.03443	0.76190	0.03115	0.75439	0.00055	0.00000	0.96120
Glass	0.56636	0.74917	0.14579	0.56410	0.23551	0.50397	0.05234	0.57143	0.65514
Image	0.92234	0.96968	0.03801	0.67654	0.03307	0.63351	0.00658	0.63158	0.94519
Iris	0.94667	0.97324	0.03600	0.66667	0.01733	0.69231	0.00000	0.00000	0.95733
Landsat	0.90651	0.93194	0.03972	0.61972	0.04572	0.57648	0.00805	0.43243	↑ 0.89927
Nursery	0.80611	0.99001	0.05622	0.85205	0.07293	0.67626	0.06474	0.55614	0.93128
Optdig	0.97737	0.98955	0.01142	0.77570	0.00961	0.73333	0.00160	0.55556	↑ 0.98395
Pendig	0.98914	0.99128	0.00615	0.84320	0.00428	0.74894	0.00044	0.50000	0.98912
Vehicle	0.52506	0.82530	0.17021	0.59167	0.24161	0.49413	0.06312	0.39700	↑ 0.67849
Wine	0.95393	0.97055	0.02135	0.89474	0.02472	0.81818	0.00000	0.00000	↑ 0.96517

other hand we consider remarkable the fact that few cases pass the filters arriving to the $K = 1$ last step. For example when the threshold is more hard, 0.8, only 4 databases arrive to the last step with more the %30 cases to be classified. And when the threshold is more flexible fewer cases arrive to K=1 classification. This means that most cases are classified with a minimum of certainty.

We do not think it is entirely fair to compare the results of the 9 *K* with the results of the 3 thresholds. As the *K*-NN has more options is more likely to get the best result. One option to make it more fair, could be to compare the average of all *K*, but neither seems to be the most appropriate. So, although the comparison seems too hard, we decide to compare with the 3 best *K* values average. In Table 5.2 we show the comparison of this two means with the mean of the different thresholds. Moreover we show the corresponding standard deviation.

Cuadro 6. Mean Accuracy level percentage and standard deviation obtained for all the Nearest Neighbors, 3 best Nearest Neighbors and our method.

	9 K		3 Best K		Our approach	
Database	Mean	σ	Mean	σ	Mean	σ
Annealing	0.90124	0.02330	**0.92739**	0.00952	0.90668	0.01386
Cmc	0.45629	0.01510	**0.46916**	0.00070	0.44177	0.00968
Dermatology	0.95804	0.00499	**0.96321**	0.00032	0.96029	0.00083
Glass	0.63551	0.01953	0.65701	0.01621	**0.67165**	0.01592
ImgSeg	0.94178	0.00647	0.94906	0.00562	**0.94993**	0.00421
Iris	0.95348	0.00523	**0.95778**	0.00077	0.95156	0.00601
Landsat	0.89471	0.00271	0.89757	0.00169	**0.90039**	0.00104
Nursery	0.92665	0.02111	**0.94259**	0.00103	0.91160	0.01796
Optdig	0.98175	0.00151	0.98335	0.00081	**0.98466**	0.00063
Pendig	0.98924	0.00184	**0.99116**	0.00090	0.99059	0.00129
StatlogVe	0.66819	0.00558	0.67392	0.00134	**0.67786**	0.00059
Wine	0.94906	0.00795	0.95543	0.00065	**0.96030**	0.00425

Looking to the results of Table 5.2 it can be seen that the mean of *DK*-NN gets better results than the mean of all the *K*, but aforementioned we regard that this is not the fairest comparison. Instead, when one compare with the 3 best *K*, *DK*-NN gets better results in 6, while in other 6 gets worst results. If the databases with small standard deviation are taken into account, one could see that our method outperform in 5 of the 8 regular databases.

In view of all these results, it seems that our proposal shows competitive behavior, getting very promising results for some databases. For example there are 4 databases where for all the thresholds, *DK*-NN gets better results than the best *K* of *K*-NN method. Furthermore when we compare the mean values our method gets better results in the %50 of the datasets.

6. Conclusion and Further Results

A new method extending the *K*-NN idea is presented in this work: *DK*-NN. The main reason of this approach is to ensure that the final decision is made with some confidence.

For doing that we find a *K* which shows a minimum of certainty in making the final decision.

The new method has been implemented and tested over 12 databases from the UCI repository. We have given different thresholds to the different *K* points and we have compared the results with those obtained by using the *K*-NN method. We do not expect our method to be better than the *K*-NN one in all the classification problems but it gets very interesting results for some databases, especially for the databases which have a regular results for different *K* in *K*-NN method.

As future work it would be interesting to change the way we assign the threshold. One option could be that the threshold indicate the minimum difference required between the two most voted class.

Referencias

1. Mitchell T.: Machine Learning. McGraw-Hill (1997)
2. Fix, E., Hodges Jr, J.L.: Discriminatory analysis, nonparametric discrimination. Technical Report Project 21-49-004, USAF school of Aviation Medicine, Randolf field (1951)
3. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification. IEEE Trans. IT-13, vol. 1, pp. 21-27 (1967)
4. Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Recognition Classification Techniques. IEEE Computer Society Press (1991)
5. Weinberger, K.Q., Saul, L.K.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. The Journal of Machine Learning Research. vol. 10, pp. 207-244 (2009)
6. Tan, S.: Neighbor-Weighted K-Nearest Neighbor for Unbalanced Text Corpus. Expert Systems with Applications. vol. 28, pp. 667-671 (2005)
7. Cost, S., Salzberg, S.: A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning, vol. 10, num. 1 , pp. 57-78 (1993)
8. Stanfill, C., Waltz, D.: Toward Memory-Based Reasoning. Communications of the ACM, vol. 29, num. 12, pp. 1213-1228 (1986)
9. Tahir, M.A., Bouridane, A., Kurugollu, F.: Simultaneous Feature Selection and Feature Weighting Using Hybrid Tabu Search/K-Nearest Neighbor Classifier. Pattern Recognition Letters archive, vol. 28, pp. 438-446 (2007)
10. MartÃnez-Otzeta, J.M., Sierra, B.: Analysis of the Iterated Probabilistic Weighted k-Nearest Neighbor Method, a New Distance-Based Algorithm. 6th International Conference on Enterprise Information Systems (ICEIS), vol. 2, pp. 233-240 (2004)
11. Sierra, B., and Lazkano, E.: Probabilistic-Weighted k Nearest Neighbor Algorithm: a New Approach for Gene Expression Based Classification. KES02 proceedings (IOS press), pp. 932-939 (2002)
12. Aha, D.: Tolerating, Irrelevant and Novel Attributes in Instance-Based Learning Algorithms. International Journal of Man-Machine Studies, vol. 36, pp. 267-287 (1992)
13. Wettschereck, D.: A Study of Distance-Based Machine Learning Algorithms. Ph.D. Thesis, Oregon State University (1994)
14. Aha, D., Kibler, D., Albert, M.K.: Instance-Based learning algorithms. Machine Learning, vol. 6, pp. 37-66 (1991)
15. Fayed, H.A., Atiya, A.F.: A Novel Template Reduction Approach for the K-Nearest Neighbor Method. IEEE Transactions on Neural Networks, vol. 20, num. 5, pp. 890-896 (2009)
16. Wang, Q., Kulkarni, S.R., Verdu, S.: Divergence Estimation for Multidimensional Densities Via K-Nearest-Neighbor Distances. IEEE Transactions on Information Theory, vol. 55, pp. 2392-2405 (2009)

17. Liaw, Y.C., Leou, M.L., Wu, C.M.: Fast Exact K Nearest Neighbors Search Using an Orthogonal Search Tree. Pattern Recognition, vol. 43, pp. 2351-2358 (2010)
18. Younes, Z., Abdallah, F., DenÅ“ux, T.: Multi-Label Classification Algorithm Derived from K-Nearest Neighbor Rule with Label Dependencies. 16th European Signal Processing Conference (2008)
19. Hellman, M.: The Nearest Neighbor Classification Rule with a Reject Option. IEEE Transactions on Systems, Man and Cybernetics, vol. 2, pp. 179-185 (1970)
20. Rodriguez, C., Boto, F., Soraluze, I., Perez, A.: An incremental and hierarchical k-NN classifier for handwritten characters. In: Proc. 16th Internat. Conf. on Pattern Recognition, Quebec City, Canada, vol. 3, pp. 98-101 (2002)
21. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update; SIGKDD Explorations, vol. 11 (2009)
22. Frank, A., Asuncion, A.: UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, Irvine, CA: University of California, School of Information and Computer Science (2010)
23. Demsar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, vol. 7, pp. 1-30 (2006)

Part III

Article Related to Multi-Classifiers

12

Classifier Subset Selection to Construct Multi-Classifiers by means of Estimation of Distribution Algorithms

Title: Classifier Subset Selection to Construct Multi-Classifiers by means of Estimation of Distribution Algorithms

Authors: I. Mendialdua, A. Arruti, B. Sierra, E. Lazkano and E. Jauregi.

Journal: Neurocomputing.

Publisher: Elsevier

Year: 2015

DOI: http://dx.doi.org/10.1016/j.neucom.2015.01.036

Neurocomputing 157 (2015) 46–60

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Classifier Subset Selection to construct multi-classifiers by means of estimation of distribution algorithms

Iñigo Mendialdua [a], Andoni Arruti [b,*], Ekaitz Jauregi [a], Elena Lazkano [a], Basilio Sierra [a]

[a] *Department of Computer Science and Artificial Intelligence, University of the Basque Country (UPV/EHU), Spain*[1]
[b] *Department of Computer Architecture and Technology, University of the Basque Country (UPV/EHU), Spain*

A R T I C L E I N F O

Article history:
Received 16 January 2014
Received in revised form
19 November 2014
Accepted 17 January 2015
Communicated by A.M. Alimi
Available online 24 January 2015

Keywords:
Machine Learning
Multiple-Classifier Systems
Evolutionary computation
Classifier Subset Selection

A B S T R A C T

This paper proposes a novel approach to select the individual classifiers to take part in a Multiple-Classifier System. Individual classifier selection is a key step in the development of multi-classifiers. Several works have shown the benefits of fusing complementary classifiers. Nevertheless, the selection of the base classifiers to be used is still an open question, and different approaches have been proposed in the literature. This work is based on the selection of the appropriate single classifiers by means of an evolutionary algorithm. Different base classifiers, which have been chosen from different classifier families, are used as candidates in order to obtain variability in the classifications given. Experimental results carried out with 20 databases from the UCI Repository show how adequate the proposed approach is; Stacked Generalization multi-classifier has been selected to perform the experimental comparisons.

1. Introduction

The Machine Learning (ML) research area, and more specifically supervised classification, addresses the problem of building, from correctly classified datasets, models able to deal with new unclassified cases and assign them a predicted class; it is worth mentioning that good classification accuracy is expected from the classifier.

It has been experimentally observed that the construction of a perfect classifier, using a single paradigm, is often impossible. Therefore, designers have tried to combine several classifiers, and this idea has led to the development of Multiple-Classifier Systems (MCS), which attempt to combine the advantages of different individual classifiers, sometimes built using different training paradigms, to obtain better results. Classifier combination is a viable alternative to using a single classifier, and has become an established research area, thriving mostly on heuristic solutions. Some theoretical results are also available but only for special cases, usually assuming independent classifier outputs [1–4].

Intuitively, it makes sense that a combination of classifiers provides better results than a single decision maker. However, this depends on how independent and diverse the individual classifiers are [5], and thus the diversity among the selected classifiers is one of the key design features within a successful multi-classifier.

There are different MCS strategies. Some approaches organize the different classifiers in a tree [6], other approaches create various classifiers by using different subsets of features to train them [7], and certain approaches divide the multi-class classification problems into several two-class sub-problems, in the so-called Class-Binarization strategies [8,9]. The most common strategies found in the literature are Boosting [10], Bagging [11] and Stacked Generalization (SG) [12].

In Bagging and Boosting, diversity is achieved by manipulating the training examples in order to generate multiple hypotheses. The base classifier is trained several times, each time with a different subset of the training examples, thus creating different classifiers. Finally there should be a method to combine the outputs of this set of classifiers.

Stacked Generalization [13] basically follows a layered architecture. At the level-0, classifiers are trained using the original dataset and each classifier outputs a prediction for each token. Successive layers receive as inputs the predictions of the previous layer, and at the level-1, a single classifier, also called meta-classifier, outputs the final prediction. The overall performance not only depends on the individual classifiers used at the level-0, but also on the correct selection of classifiers at other levels. One problem of Stacked Generalization is how to obtain the right combination of level-0 classifiers and the meta-classifier, especially in relation to each specific dataset.

* Corresponding author.
E-mail address: andoni.arruti@ehu.es (A. Arruti).
[1] http://www.sc.ehu.es/ccwrobot

http://dx.doi.org/10.1016/j.neucom.2015.01.036
0925-2312/© 2015 Elsevier B.V. All rights reserved.

Two difficulties arise when a MCS has to be developed: the selection of base classifiers, and how to combine their individual decisions.

By combining the outputs of a team of classifiers, we aim at a more accurate decision than that of the best member of the team. The assumption is that developers should introduce diversity in the ensemble, and therefore enhance the performance. There are different ways to tackle output combination; a weighted majority vote is the standard combination method for ensembles but there are other combination methods which could also be successful. Many combination methods and algorithms have been developed, including the use of a meta-classifier, which is the option selected to construct a Stacked Generalization model.

The difficulty in choosing a suitable combination method for the problem at hand has been recognized and highlighted numerous times in the literature. Some theoretical works rely on simplifications and assumptions, and consider mostly special cases [14–16]. However, even a discipline as mature as pattern recognition does not offer strict guidelines about how to approach a dataset and which classifier to select for it; many experimental studies have been published in the search for such guidelines [17,18]. This study also belongs to this experimental group.

This paper presents a methodology to incorporate into a Multiple-Classifier System a mechanism which attempts to adapt the structure of the MCS to a given classification problem. This approach has the following properties:

1. To be a global optimization technique which has been shown to be successful in complex domains, such as the space of the possible configurations for a Multiple-Classifier System given a pool of individual classifiers.
2. To provide an easy way to express the problem of optimization of the multiple-classifier structure.
3. To combine the individual classifier opinions in the form established by the MCS itself (voting, meta-classifier, etc.); the searching process itself selects the appropriate base classifiers taking into account the mode in which the final decision is taken.

We present a Multiple-Classifier System which incorporates an automatic self-configuration scheme based on Estimation of Distribution Algorithms [19] (an evolutionary computation approach). Our main interest is focused on the constituent classifiers of the resulting multi-classifier.

To show the behaviour of the proposed method, 20 datasets have been selected from the UCI repository [20] and 10 standard classifiers are used by the new Classifier Subset Selection (CSS) proposed method to construct a Stacked Generalization MCS. Experiments are carried out comparing the results of the 10 single classifiers, state-of-the-art multi-classifiers (Boosting, Bagging and StackingC) and the Stacked Generalization classifiers constructed with the whole set of 10 base classifiers in the level-0. Results obtained show the goodness of the approach, as the new paradigm statistically outperforms the remaining classifiers used.

The rest of the paper is organized as follows: in Section 2 related work in the area of multi-classifiers is presented. Section 3 describes the proposed approach and Section 4 the experimental setup. Finally, Section 5 depicts some conclusions and future work lines.

2. Related work

Several papers can be found in the literature about construction and use of Multiple-Classifier Systems. In this section we reflect three main aspects: different MCS which have been used in different tasks; in the next subsection the revision is focused on the Stacked Generalization paradigm, as it will be that used in the experimental phase, and in the third subsection we revise the use of evolutionary algorithms in order to obtain better multi-classifiers.

2.1. Multiple-Classifier Systems

Combination of classifiers has been widely used as a useful approach in several Machine Learning tasks [21].

In the field of people detection, several authors have used multi-classifier approaches: [22] uses Histograms of Oriented Gradients (HOGs) and Local Receptive Fields (LRFs), which are provided by a convolutional neural network, and are classified by Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) combining classifiers by majority vote and fuzzy integral; [23–25] present a MCS to manage image based classification problems; Batista et al. [26] take advantage of unigrams, bigrams and trigrams to design a Multiple-Classifier System for Sentiment Analysis and Opinion Mining.

In [27], to improve the performance of classification, three classifiers that have the best results among all applied methods are combined; on the other hand, [28] presents a Multiple-Classifier System based on colour and texture information for face image segmentation; Haibo et al. [29] present a hybrid MSC to improve the precision of remote sensing image classification. Taking the characteristic of abstract level and measurement level into consideration, the optimal sub-classifier, Bagging algorithm and the largest confidence algorithm are combined.

In [30], the problem of Multiple-Classifier System design is discussed and the reader is provided with a critical survey of the state-of-the-art. The main conclusion in this section is that optimal design is still an open problem. More information about Multiple-Classifier Systems can be found in [31].

2.2. Stacked Generalization

Ting and Witten [32] propose to extend Stacked Generalization using class probability distributions of the original classifiers. Moreover, they propose to use the Multi-Response Linear Regression (MRLR) as meta-classifier. Seewald [33] discovered that this new version worked correctly for two-class problems while it performed worse for multi-class problems. In order to solve this problem, he proposes a new method called StackingC where, for each class separately, a meta training set is created with the class probabilities associated with the class. In this case, he also uses MRLR as meta-classifier.

As can be seen in the literature the Stacked Generalization multi-classifier has been applied in different types of problems: For example Ekbal et al. [34] use a Stacked Generalization multi-classifier for the extraction of biomedical entities in the forms of genes and gene product mentions in text and Ibarguren et al. [35] use a Stacked Generalization to real time recognition of the Fingerspelling Alphabet used by the deaf people.

2.3. Base classifier configuration by means of evolutive computation

Quian et al. investigate MSC as a multi-goal problem from a theoretical point of view [36]. Whereas Rahman et al. [37] propose a novel cluster oriented ensemble classifier generation method and a Genetic Algorithm based approach to optimize the parameters.

Impedovo et. al. [38] propose a new method for handwritten digit recognition where, for each individual classifier, a feature selection is applied. They consider the problem of feature selection as an optimization problem so they use the genetic algorithms in order to find the best performance of the combined classifier. On the other hand, in order to predict subcellular localization of apoptosis proteins, Ding and Zhang [39] propose a new method where each individual classifier is trained with different dimensions of protein sequences and they use the genetic algorithms to find the optimal weight factors.

Zhou et al. [40] have proved that ensembling some of the available classifiers may be better than ensembling all of them. Viewing that, Kim and Kang [41] propose a new method where they use the genetic algorithms for classifier selection in ensembles. In this case they concentrate on the selection process of an ensemble containing diverse classifiers.

In the literature we can also find some works that try to improve the performance of Stacked Generalization by selecting the best classifiers using different evolutionary computation strategies. Chen [42,43] proposes a new ensemble construction method which applies Ant Colony Optimization (ACO) in the Stacked Generalization ensemble construction process to generate domain-specific configurations. Shunmugapriya and Kanmani [44], use Artificial Bee Colony (ABC) Algorithm as a meta-heuristic search algorithm to obtain a suitable Stacked Generalization model. To this end, two versions of the ABC algorithm are used. Ledezma et al. [45] use the genetic algorithms and show that selecting the right classifiers, their parameters and the meta-classifier is a critical issue.

3. Proposed approach

As explained in the Introduction, the main goal of this work is optimal selection of base classifiers to construct multi-classifiers. In this section, the elements and processing sequence that constitute the proposed approach are explained in detail.

3.1. Combination of classifiers

To combine the results of the base classifiers, we use Stacked Generalization (SG) as Multiple-Classifier System. Stacked Generalization is a well known ensemble approach and is also called Stacking [12,46]. While ensemble strategies such as Bagging or Boosting obtain the final decision after a vote among the predictions of the individual classifiers, SG applies another individual classifier to the predictions in order to detect patterns and improve performance of the vote.

As can be seen in Fig. 1, SG is divided into two levels: in the level-0 each individual classifier makes a prediction independently, and in the level-1 these predictions are treated as the input values of another classifier, known as meta-classifier, which returns the final decision.

The data for training the meta-classifier is obtained after a validation process, where the outputs of the level-0 classifiers are taken as attributes and the class is the real class of the example.

3.2. Classifier Subset Selection (CSS)

Although using many classifiers may seem more effective, our believe is that selecting a subset of them can reduce the computational cost and improve the accuracy, assuming that the selected classifiers are diverse and independent between them.

In this paper we propose a new multi-classifier paradigm, which extends the Staking Generalization approach, reducing the number of classifiers to be used in the final model. We call this new approach Classifier Subset Selection (CSS) and a graphical example is illustrated in Fig. 2. As can be seen, we added to the multi-classifier an intermediate phase in which a subset of the level-0 classifiers is selected. The criterion to make the selection depends on the goal of the classification task, and in this case, we have decided to use classification accuracy. As can be seen in Fig. 2, discarded classifiers – those with an $\times$ – are not used in the multi-classifier.

3.3. CSS by means of estimation of distribution algorithms (EDAs)

CSS can be contemplated in a similar way as Feature Subset Selection (FSS) in some ML problems. As reported by Aha and Bankert [47], the objective of FSS in Machine Learning is to reduce the number of features used to characterize a dataset so as to

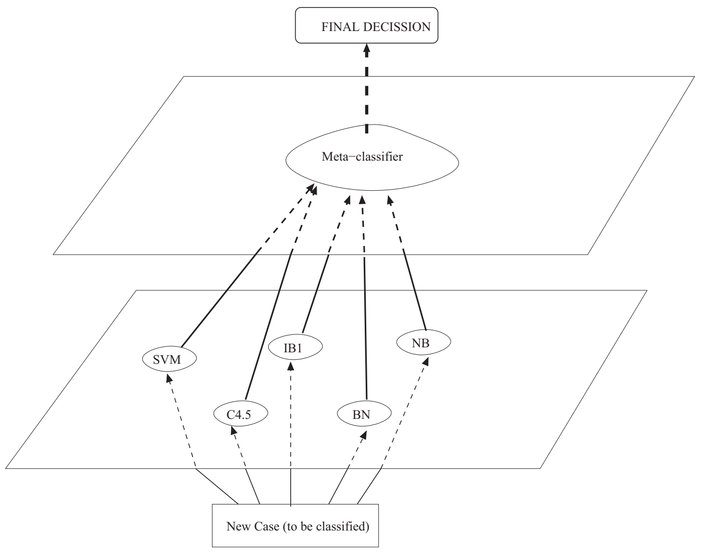


Fig. 1. Stacked Generalization schemata.

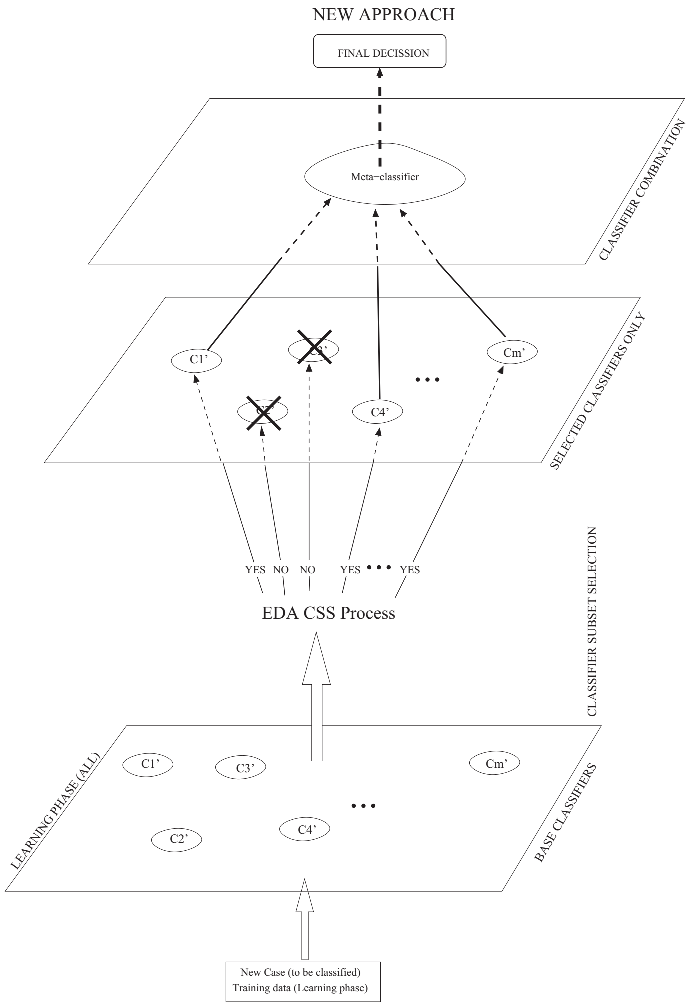


Fig. 2. Classifier Subset Selection Stacked Generalization.

improve the performance of a learning algorithm on a given task. Our objective will be the maximization of the classification accuracy in a multi-classifier; CSS task can be thus exposed as a search problem, each state in the search space identifying a subset of possible base classifiers selected.

The method used to select the classifiers could be any one, but in this type of scenarios evolutionary approaches are often introduced with promising results. Today, some of the best known evolutionary algorithms for FSS are based on Estimation of Distribution Algorithms (EDAs) [19]. EDA combines statistical learning with population-based search in order to automatically identify and exploit certain structural properties of optimization problems. Inza et al. [48] proposed an approach that used an EDA called Estimation of Bayesian Network Algorithm (EBNA) [49] for a FSS problem. Viewing that in [50] EBNA shows better behaviour than genetic and sequential search algorithms for FSS problems (and hence for CSS in this approach), we decide to use EBNA. Moreover EBNA has been selected as the model in the recent work that analyses the behaviour of the EDAs [51].

EBNA is an EDA that learns a Bayesian Network and it follows the typical EDA structure. It starts with a population of candidate solutions to the problem, starting with a population generated with uniform distribution over all admissible solutions. The population is then scored using a fitness function. This fitness function gives a numerical ranking for each string, with the higher the number, the better the string. From this ranked population, a subset of the most promising solutions are selected by the selection operator. The characteristic of EBNA is that it uses a Bayesian Network to deal with the probability distribution of the selected solutions. Once the model is constructed, new solutions are generated by sampling the distribution encoded by this model. These new solutions are then incorporated back into the old population. The process is repeated until some termination criteria are met (usually when a solution of sufficient quality is reached or

```
EBNA
    $D_{t=0} \leftarrow$ Generate $N$ individuals (the initial population) randomly.
    Repeat for $l = 1, 2, \ldots$ until a stop criterion is met.
    $V_i \leftarrow$ Evaluate the Value $V_i$ for each of the $N$ individuals.
        $D^s_{t-1} \leftarrow$ Select $S \leq N$ individuals from $D_{t-1}$ according to a selection method.
        $p_l(\mathbf{x}) = p(\mathbf{x}|D^s_{l-1}) \leftarrow$ Estimate the joint probability distribution of an individual
        being among the selected individuals with the trained Bayesian Network.
        $D_l \leftarrow$ Sample $N$ individuals (the new population) from $p_l(\mathbf{x})$.
```

Fig. 3. Main scheme of the EDA approach.

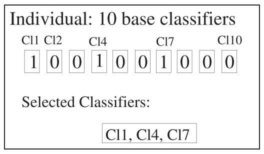


Fig. 4. The combinations of base classifiers as EDA individuals.

when the number of iterations reaches some threshold). Fig. 3 shows the pseudocode of the algorithm.

In our approach, an individual in the EDA algorithm will be defined as an n-tuple of binary 0, 1 values – the so-called Binary Encoding – and each position in the tuple refers to a concrete base classifier, and the value indicates whether this classifier is used (1 value) or not (0 value). An example with 10 classifiers (the value used in this paper) can be seen in Fig. 4. In this example, Cl1, Cl4 and Cl7 are the selected classifiers, and the remaining seven are not used.

Once an individual has been sampled, it has to be evaluated. The aim is to consider the predictive power of each subset of base classifiers. For this, a multi-classifier is built for each individual using the corresponding subset of classifiers and the obtained validated accuracy is used as a fitness function. Thus, when looking for the individual that maximizes the fitness function, the EDA algorithm is also searching the optimal subset of base classifiers.

For individual selection we decide to use the range-based selection, selecting the best S individuals from the N individuals of the population. Although S can be any value, we decide to set it to $N/2$ since it is the most common value in EDA literature.

4. Experimental results

In this section we show the set-up of the experimental framework and the obtained results.

4.1. Datasets

In order to evaluate the performance of the proposed approach, 20 databases have been selected from the UCI repository [20]. In Table 1 the characteristics of these databases are shown. The number of cases ranges from 101 to 1728, the number of attributes from 4 to 35 and the number of classes from 2 to 19, so a wide variety of problems is represented.

Table 1
Characteristics of the databases.

Database	#Cases	#Classes	#Attributes
Balance-scale	625	3	4
Breast-cancer	286	2	9
Car	1728	4	6
Cmc	1473	3	9
Colic	368	2	22
Diabetes	768	2	8
Ecoli	336	8	7
Glass	214	7	9
Hepatitis	155	2	19
Iris	150	3	4
Lymph	148	4	18
Liver-disorders	345	2	6
Solar-flare-1	323	6	12
Solar-flare-2	1066	6	12
Soybean	683	19	35
Vehicle	846	4	18
Vote	435	2	16
Vowel	990	11	13
Wine	178	3	13
Zoo	101	7	17

4.2. Base classifiers

To carry out the experiments, we have used 10 well-known ML supervised classification algorithms from a software package for Machine Learning called WEKA [52].

Among the classifiers that WEKA offers, we have selected the following:

1*R*: is a one level decision tree which tests just one attribute [53]. The chosen attribute is that which produces minimum error.

KNN: is a case-based, Nearest-Neighbour classifier [54]. To classify a new test sample, a simple distance measure is used to find the training instance closest to the given test instance, and then it predicts the same class as this nearest training instance.

RIPPER: (Repeated Incremental Pruning to Produce Error Reduction) [55] is a rule-base learner, an optimized version of IREP, that forms rules through a process of repeated growing (to fit training data) and pruning (to avoid overfitting). RIPPER handles multiple classes by ordering them from least to most prevalent, and then treating each in order as a distinct two-class problem.

Naive Bayes (*NB*): The Naive-Bayes rule [56] uses the Bayes theorem to predict the class for each case, assuming that the predictive genes are independent given the category. To classify a new sample characterized by d genes $\mathbf{X} = (X_1, X_2, \ldots, X_d)$, the NB classifier applies the following rule:

$$c_{NB} = \arg\max_{c_j \in C} p(c_j) \prod_{i=1}^{d} p(x_i \mid c_j)$$

where c_{NB} denotes the class label predicted by the Naive-Bayes classifier and the possible classes of the problem are grouped into $C = \{c_1, \ldots, c_l\}$.

*C*4.5: The C4.5 [57] represents a classification model by a decision tree. The tree is constructed in a top-down way, dividing the training set and beginning with the selection of the best variable in the root of the tree.

*K**: is an instance-based algorithm that uses an entropy-based distance function [58].

Table 2
Single classifiers' results.

Dataset	1R	KNN	RIPPER	NB	C4.5	K*	BN	NBT	RF	SVM
Balance-scale	56.32	86.56	80.80	**90.40**	76.64	88.48	72.32	76.64	80.48	87.68
Breast-cancer	65.73	72.38	70.98	71.68	**75.53**	73.43	72.03	70.98	69.23	69.58
Car	70.02	93.52	86.46	85.53	92.36	87.56	85.71	**94.21**	92.65	93.75
Cmc	48.13	44.33	**52.41**	50.78	52.14	50.24	51.05	51.73	50.85	48.20
Colic	81.52	81.25	84.24	77.99	85.33	76.63	81.25	82.06	**86.14**	82.61
Diabetes	72.79	70.18	76.04	76.30	73.83	69.14	74.35	74.35	73.83	**77.34**
Ecoli	64.88	80.36	81.25	**85.42**	84.23	80.95	81.25	82.14	83.63	84.23
Glass	58.41	70.56	68.69	48.60	66.82	**75.23**	70.56	70.56	72.90	56.08
Hepatitis	83.23	80.64	78.06	84.52	83.87	81.94	83.23	80.00	82.58	**85.16**
Iris	93.33	95.33	94.00	**96.00**	**96.00**	94.67	92.67	94.00	95.33	**96.00**
Liver-disorders	59.42	62.90	64.64	55.36	68.70	66.96	56.23	66.09	**68.99**	58.26
Lymph	75.00	82.43	77.70	83.11	77.03	85.14	85.81	80.41	81.08	**86.49**
Solar-flare-1	54.80	68.11	**72.45**	65.64	72.14	69.35	66.87	70.59	68.42	70.28
Solar-flare-2	61.45	72.89	70.45	74.39	74.48	74.67	74.48	74.58	72.98	**75.23**
Soybean	39.97	91.22	91.95	92.97	91.51	87.99	93.27	91.51	91.66	**93.85**
Vehicle	51.42	69.86	68.56	44.80	72.46	71.39	60.05	72.93	**77.07**	74.35
Vote	95.63	92.41	95.40	90.11	**96.32**	93.33	90.11	95.63	95.86	96.09
Vowel	31.82	**99.29**	69.70	63.74	81.52	98.99	60.81	93.53	96.06	71.41
Wine	77.53	94.94	91.57	96.63	93.82	**98.88**	**98.88**	96.63	97.19	98.31
Zoo	42.57	**96.04**	86.14	95.05	92.08	**96.04**	94.06	94.06	89.11	**96.04**

Table 3
Bagging, Boosting and StackingC results.

Dataset	AdaBoost M1 (DS)	Bagging (REPTree)	AdaBoost M1 (C4.5)	Bagging (C4.5)	StackingC
Balance-scale	72.32	82.72	78.88	82.24	**88.96**
Breast-cancer	70.28	68.88	69.58	**73.43**	73.08
Car	70.02	91.67	**96.12**	93.52	95.54
Cmc	42.70	**54.24**	50.78	54.11	53.77
Colic	81.25	84.78	83.42	**85.60**	84.51
Diabetes	74.35	74.48	72.40	74.09	**76.17**
Ecoli	64.58	83.33	81.25	84.82	**86.01**
Glass	44.86	71.03	74.30	71.03	**74.77**
Hepatitis	82.58	83.23	**85.81**	83.23	83.87
Iris	**95.33**	94.67	93.33	**95.33**	93.33
Liver-disorders	66.09	71.30	71.59	**72.75**	70.72
Lymph	74.32	79.05	81.08	79.05	**85.14**
Solar-flare-1	46.75	**73.38**	71.21	72.14	72.75
Solar-flare-2	53.47	74.86	73.17	73.83	**75.42**
Soybean	27.97	87.12	92.83	93.27	**93.70**
Vehicle	39.95	72.34	76.24	76.60	**76.83**
Vote	95.40	95.86	95.86	**96.32**	96.09
Vowel	17.37	86.87	93.33	90.40	**99.09**
Wine	91.57	94.94	96.63	94.94	**97.75**
Zoo	60.40	42.57	**95.05**	93.07	**95.05**

Table 4
Stacked Generalization results.

Dataset	1R	KNN	RIPPER	NB	C4.5	K*	BN	NBT	RF	SVM
Balance-scale	93.12	90.56	92.96	92.48	93.28	89.92	91.68	93.92	**94.72**	92.80
Breast-cancer	69.23	65.03	69.23	69.23	71.68	65.03	69.93	72.03	70.28	**72.73**
Car	93.92	96.70	99.07	95.72	98.90	97.22	96.70	98.21	**99.31**	98.03
Cmc	45.21	45.48	50.85	53.90	47.86	47.73	53.56	52.75	52.48	**54.11**
Colic	**85.60**	78.26	84.51	85.05	82.34	79.89	**85.60**	82.88	84.51	84.24
Diabetes	71.48	70.18	74.87	76.56	76.17	71.22	74.74	75.00	74.22	**77.08**
Ecoli	72.32	80.95	83.33	74.70	82.74	80.66	83.63	82.44	**84.23**	53.87
Glass	56.08	66.82	74.30	62.62	71.96	66.82	73.36	65.42	**75.70**	66.82
Hepatitis	81.29	80.64	81.94	**85.16**	83.23	81.29	83.23	84.52	83.87	84.52
Iris	92.00	93.33	92.67	95.33	94.67	**96.00**	95.33	94.67	94.67	93.33
Liver-disorders	62.90	59.42	66.96	**71.30**	64.06	58.84	70.72	68.12	66.67	70.44
Lymph	81.08	79.05	**84.46**	81.76	83.78	81.76	83.11	79.73	81.76	**84.46**
Solar-flare-1	61.30	65.94	66.25	58.51	68.73	69.97	65.64	64.40	67.80	**70.90**
Solar-flare-2	64.54	70.73	72.05	71.95	71.11	70.36	71.58	70.36	71.39	**74.86**
Soybean	45.68	93.12	90.34	92.24	90.78	6.04	**93.85**	91.95	93.27	**93.85**
Vehicle	54.26	73.88	77.19	75.06	76.60	75.77	78.01	74.94	77.42	**78.49**
Vote	95.86	95.40	95.40	96.32	96.09	95.86	96.09	95.40	**97.01**	96.09
Vowel	34.75	99.09	97.37	98.18	98.38	96.97	98.08	94.95	98.89	**99.29**
Wine	92.70	98.88	98.88	97.19	98.88	98.31	98.88	**99.44**	98.88	97.75
Zoo	73.27	**97.03**	95.05	96.04	92.08	**97.03**	93.07	89.11	93.07	94.06

Bayesian Networks (*BN*): A Bayesian network [59], belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG).

Naive Bayes Tree (*NBT*): uses a decision tree with naive Bayes classifiers at the leaves [60].

Random Forest (*RF*): constructs a combination of many unpruned decision trees [61]. The output class is the mode of the classes output by individual trees.

Support Vector Machines (*SVM*): are a set of related supervised learning methods used for classification and regression [62]. Viewing input data as two sets of vectors in an *n*-dimensional space, an SVM will construct a separating hyperplane in that space, one which maximizes the margin between the two datasets.

As can be seen, we have selected classifiers with different approaches to learning, and widely used in different classification tasks. The goal is to combine them in a multi-classifier to maximize the benefits of each modality by intelligently fusing

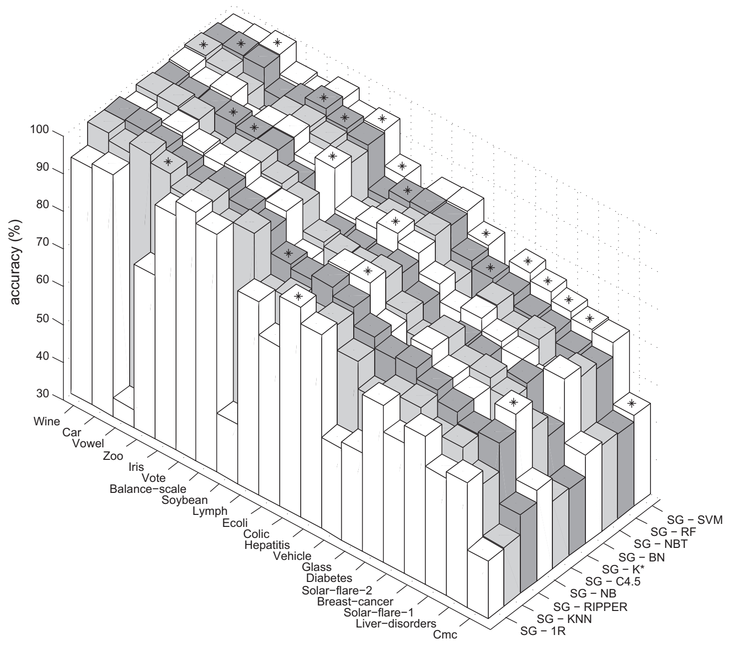


Fig. 5. Stacked Generalization results obtained for each database and meta-classifier.

Table 5
Results obtained by the proposed approach.

Dataset	1R	KNN	RIPPER	NB	C4.5	K*	BN	NBT	RF	SVM
Balance-scale	93.12	95.04	95.20	93.92	94.72	95.20	93.44	95.68	**96.48**	94.24
Breast-cancer	74.13	74.48	75.17	**75.52**	75.17	74.83	73.43	**75.52**	**75.52**	74.48
Car	93.92	99.13	99.48	98.15	99.48	99.42	98.78	99.42	**99.77**	98.67
Cmc	48.47	52.00	53.90	**55.67**	53.23	53.16	55.53	55.13	52.68	55.33
Colic	86.69	85.33	**87.77**	86.69	**87.77**	87.23	86.96	87.23	85.60	86.14
Diabetes	77.34	77.60	78.13	78.00	78.52	**78.65**	76.82	78.00	78.13	77.47
Ecoli	75.89	86.01	86.61	84.23	86.91	86.61	85.42	84.82	87.80	**88.09**
Glass	64.49	75.23	78.04	73.83	77.57	78.97	78.50	72.43	**79.91**	79.44
Hepatitis	85.81	85.16	**87.74**	86.45	**87.74**	85.81	85.81	**87.74**	**87.74**	85.16
Iris	96.00	**97.33**	96.67	96.67	96.67	**97.33**	96.67	96.67	96.67	96.67
Liver-disorders	71.01	68.70	72.46	**73.04**	71.88	71.30	71.88	71.59	70.44	72.75
Lymph	83.78	86.49	86.49	87.16	86.49	85.81	**88.51**	85.14	87.16	87.84
Solar-flare-1	63.78	**75.23**	73.06	70.59	73.99	74.30	73.06	73.38	73.99	73.99
Solar-flare-2	66.04	75.05	73.64	73.45	75.23	75.33	72.89	74.39	75.33	**76.36**
Soybean	50.22	94.14	93.70	93.56	93.41	93.41	92.83	91.65	**94.44**	**94.44**
Vehicle	63.00	77.54	78.72	77.31	78.49	78.84	78.84	77.19	**80.38**	79.79
Vote	96.78	96.78	96.78	97.01	96.32	97.01	**97.24**	**97.24**	**97.24**	96.55
Vowel	37.98	**99.50**	99.29	99.09	99.29	99.19	98.28	98.79	**99.50**	99.39
Wine	92.70	**100.00**	**100.00**	**100.00**	**100.00**	99.44	**100.00**	**100.00**	98.88	**100.00**
Zoo	75.25	**98.02**	96.04	**98.02**	97.03	**98.02**	95.05	95.05	**98.02**	**98.02**

their information, and by overcoming the limitations of each modality alone. As we treat the classifiers as black boxes, we have used the default parameters of the classifiers.

4.3. Method

The experimental phase is divided into four steps that are applied to each of the 20 databases:

1. Single classifiers: build 10 classifiers, applying the 10 base Machine Learning algorithms to the training dataset, and get validated classification accuracies.
2. Standard multi-classifiers: build 5 classifiers, applying Bagging (REPTree and C4.5), Boosting (DecitionStump and C4.5) and StackingC Machine Learning algorithms to the training dataset, and get validated classification accuracies.
3. Stacked Generalization: applying classic Stacked Generalization algorithm (with the ten base classifiers at level-0), build 10

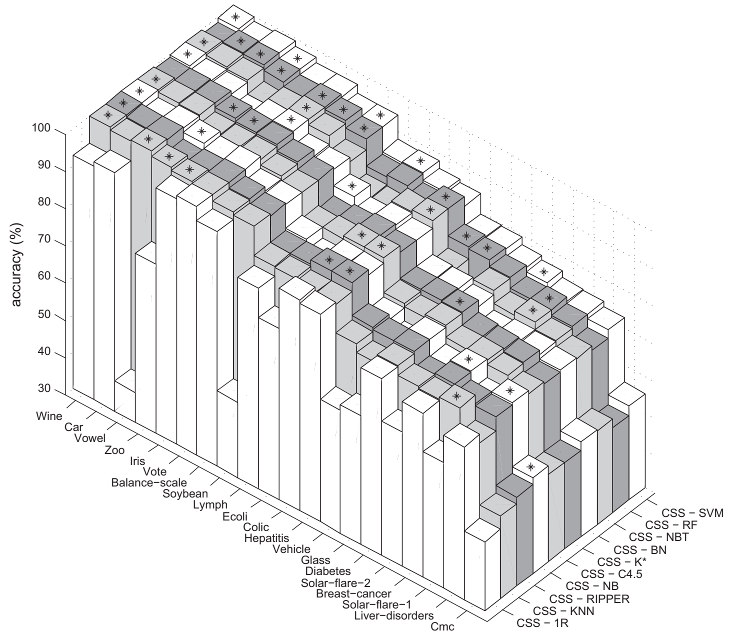


Fig. 6. Results obtained by the proposed approach for each database and meta-classifier.

Table 6
Mean and standard deviation results obtained for each of the classifier sets used: single, ensemble, Stacked Generalization and CSS.

Classifier set	Single		Ensemble		Stacking		CSS	
Dataset	Mean	StDev	Mean	StDev	Mean	StDev	Mean	StDev
Balance-scale	79.63	10.13	79.04	4.79	92.54	1.46	**94.70**	**1.04**
Breast-cancer	71.15	2.63	70.54	2.01	69.44	2.63	**74.83**	**0.70**
Car	88.18	7.30	87.83	12.01	97.38	**1.69**	**98.62**	1.72
Cmc	49.99	2.47	50.46	5.41	50.39	3.51	**53.51**	**2.20**
Colic	81.90	2.98	83.76	1.90	83.29	2.48	**86.74**	**0.84**
Diabetes	73.82	2.59	73.83	0.97	74.15	2.39	**77.86**	**0.56**
Ecoli	80.83	5.86	78.50	9.39	77.89	9.32	**85.24**	**3.50**
Glass	65.84	8.58	65.30	13.72	67.99	6.01	**75.84**	**4.70**
Hepatitis	82.32	2.20	83.71	1.43	82.97	1.59	**86.52**	**1.12**
Iris	94.73	1.19	94.67	0.94	94.20	1.30	**96.73**	**0.38**
Liver-disorders	62.75	5.11	70.43	2.97	65.94	4.52	**71.51**	**1.27**
Lymph	81.42	3.92	78.38	2.87	82.09	1.86	**86.49**	**1.35**
Solar-flare-1	67.86	5.07	65.87	12.78	65.94	3.82	**72.54**	**3.31**
Solar-flare-2	72.56	4.15	68.83	10.26	70.89	**2.58**	**73.77**	2.91
Soybean	86.59	16.46	75.29	31.68	79.11	29.60	**89.18**	**13.71**
Vehicle	66.29	10.69	66.28	17.66	74.16	7.15	**77.01**	**5.03**
Vote	94.09	2.44	95.86	0.38	95.95	0.50	**96.90**	**0.31**
Vowel	76.69	21.60	71.99	36.51	91.60	20.01	**93.03**	**19.35**
Wine	94.44	6.38	94.52	2.12	97.98	**1.97**	**99.10**	2.28
Zoo	88.12	16.34	72.77	25.65	91.98	7.01	**94.85**	**7.00**

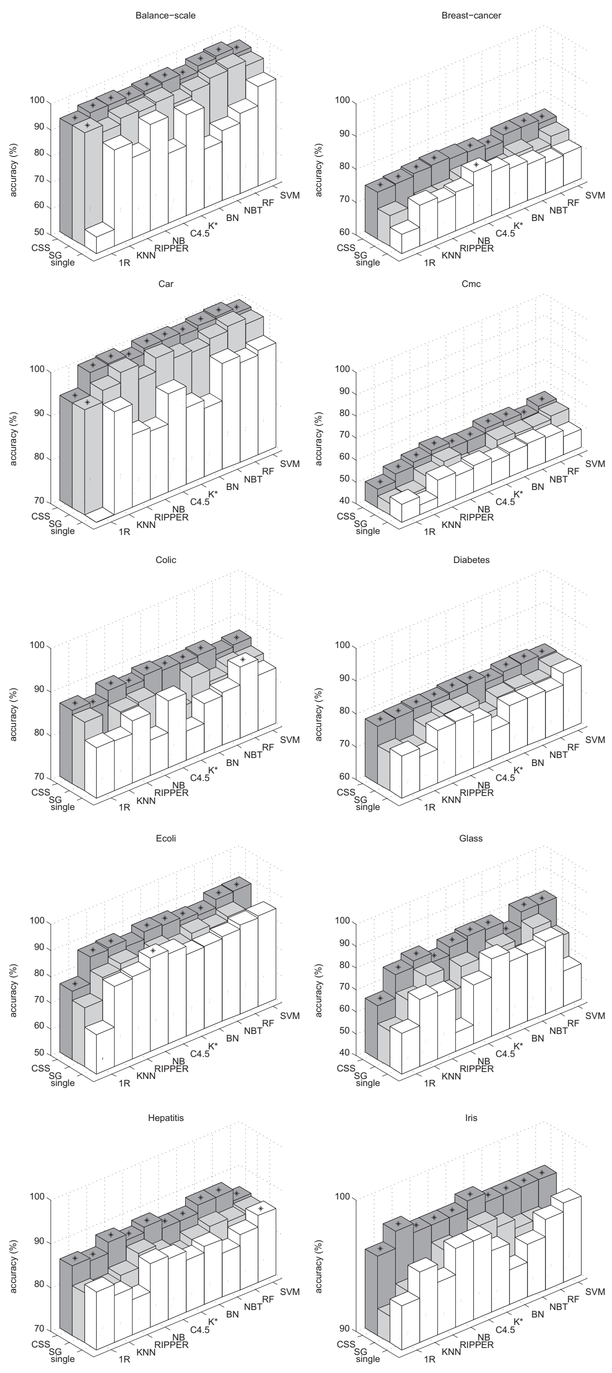


Fig. 7. Comparison of CSS, SG and single classifiers for each database with different meta-classifiers.

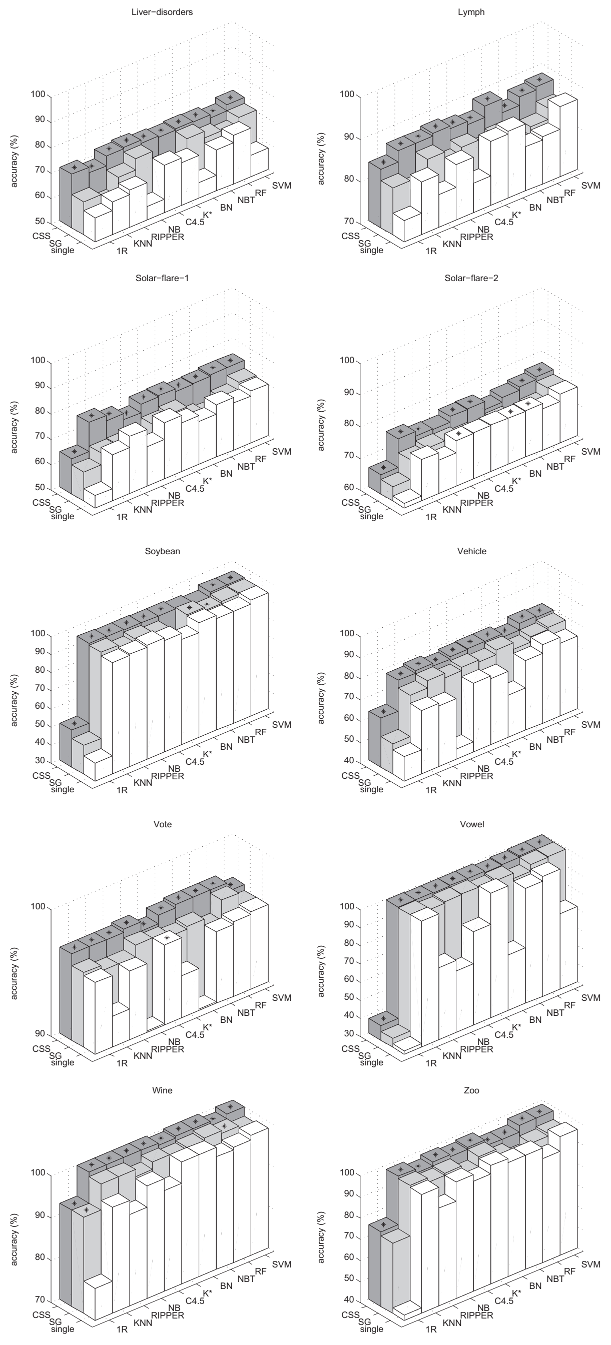


Fig. 8. Comparison of CSS, SG and single classifiers for each database with different meta-classifiers.

classifiers, one for each base classifier at level-1, and get validated classification accuracies.

4. Classifier Subset Selection for Stacked Generalization: using our new approach, build 10 classifiers, one for each base classifier at level-1, and each of them with only a subset of classifiers, selected by EDA algorithm.

4.4. *Experimental setup*

In all the experiments 10-fold cross-validation [63] has been used to get a validated classification accuracy (well classified rate), and this accuracy has been the criterion to define the fitness of an individual, inside the evolutionary algorithm.

For Classifier Subset Selection, the EDA algorithm used has been EBNA (Estimation of Bayesian Network Algorithm) [49], with Algorithm B [64] for structural learning of the Bayesian Network. Population size *N* has been set to 50 individuals, representing 50 combination of classifiers, the number *S* of selected individuals at each generation is 20 (40% of the population size), and the number of generations of new individuals was set to 4.

4.5. *Obtained results*

4.5.1. *Single classifiers' results*

All base classifiers are evaluated independently for each dataset; it is worth mentioning that the obtained results are supposed to be outperformed by multi-classifiers.

Table 2 shows the results obtained by each single classifier on each dataset. As can be seen, the type of classifier that obtains the best accuracy for each classification problem varies considerably. As a matter of fact, all of the base classifiers, except 1R, obtain the best validated result for at least one dataset.

4.5.2. *Standard multi-classifiers' results*

In order to obtain a more honest comparison with the proposed approach, some state-of-the-art multi-classifier results are also shown; we have used Bagging, Boosting and StackingC

Table 3 shows the result obtained by those standard multi-classifiers. The best results are obtained by different approaches for different datasets as well.

4.5.3. *Stacked Generalization results*

Obtained results for Stacked Generalization classifier ensembles are shown in Table 4, using the 10 base paradigms as meta-classifier (level-1) in the SG structure. All paradigms used as meta-classifier, except C4.5, obtain the best result for at least one of the classification problems. It should be noticed that the meta-classifier with the best result differs in general from the best single classifier obtained for the same dataset; only in five of the 20 datasets (diabetes, lymph, solar-flare-2, soybean and zoo) there is a coincidence.

A graphical image of the results obtained by the SG paradigm is shown in Fig. 5; it can be seen that, for each dataset, the result varies significantly for each meta-classifier. This fact could indicate that the selection of an appropriate meta-classifier is very important in the SG classifier design.

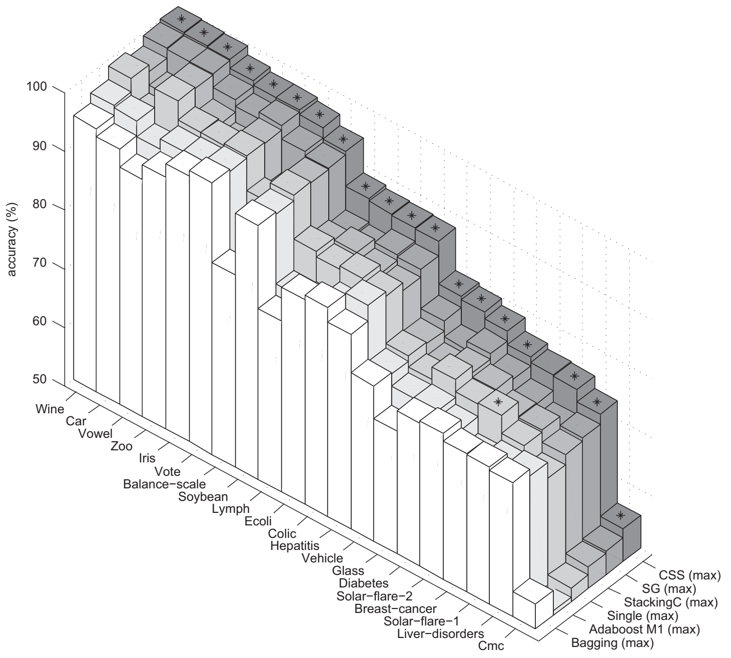


Fig. 9. Proposed approach versus best results obtained by the remaining paradigms.

Table 7
Configuration of the best multi-classifiers obtained for each dataset.

Dataset	Meta	Base classifiers
Balance-scale	RF	1R, RIPPER, NB, K*, BN, SVM
Breast-cancer	NB	1R, KNN, RIPPER, NB, K*, BN, NBT, RF, SVM
	NBT	1R, KNN, RIPPER, NB, C4.5, BN, NBT, RF
	RF	1R, KNN, RIPPER, NB, BN, RF
Car	RF	1R, KNN, NB, C4.5, K*
Cmc	NB	1R, RIPPER, C4.5, K*, NBT, SVM
Colic	RIPPER	KNN, K*, BN, NBT, RF
	C4.5	NBT, RF, SVM
Diabetes	K*	RF, SVM
Ecoli	SVM	1R, NB, K*, RF
Glass	RF	1R, KNN, C4.5, K*
Hepatitis	RIPPER	KNN, RIPPER, NB, SVM
	C4.5	RIPPER, NB, BN, NBT, SVM
	NBT	1R, KNN, RIPPER, NB, RF, SVM
	RF	1R, NB, BN, RF, SVM
Iris	KNN	1R, KNN, NB, C4.5, RF, SVM
	K*	KNN, NB, C4.5, K*, BN, SVM
Liver-disorders	NB	KNN, RIPPER, C4.5, RF, SVM
Lymph	BN	KNN, NB, C4.5, K*, BN, NBT, RF, SVM
Solar-flare-1	KNN	1R, SVM
Solar-flare-2	SVM	KNN, NB, NBT, RF, SVM
Soybean	RF	1R, RIPPER, C4.5, K*, BN, NBT, SVM
	SVM	NB, RF, SVM
Vehicle	RF	1R, NB, C4.5, K*, BN, NBT, RF
Vote	BN	C4.5, K*, NBT, RF, SVM
	NBT	KNN, RIPPER,C4.5, RF
	RF	1R, KNN, NB, C4.5, NBT, RF, SVM
Vowel	KNN	KNN, NB, C4.5, K*, SVM
	RF	KNN, NB, C4.5, K*, RF
Wine	KNN	RIPPER, NB, BN, NBT, RF
	NB	1R, RIPPER, NB, K*, NBT, RF, SVM
	C4.5	RIPPER, NB, C4.5, RF, SVM
	BN	1R, C4.5, K*, BN, RF, SVM
	NBT	1R, KNN, NB, C4.5, K*, RF, SVM
	RF	C4.5, K*, BN, RF
	SVM	1R, KNN, RIPPER, C4.5, BN, SVM
Zoo	KNN	1R, KNN, RIPPER, NB, C4.5, RF, SVM
	NB	KNN, K*, BN, NBT, SVM
	K*	KNN, RIPPER, NB, C4.5, BN, NBT, SVM
	RF	NB, C4.5
	SVM	RIPPER, NB, BN, NBT, RF

4.5.4. *Classifier Subset Selection for Stacked Generalization*

Finally CSS has been applied to each of the datasets and meta-classifiers; obtained results indicate the appropriateness of the proposed approach.

It should be noticed that, although the learning time is time consuming (until the EDA search converges), the classification time is very short, and it can be done in real time. In this way, and for any dataset used, once the classifier is constructed it can be used at a very high frequency, as it is composed of fast classifiers.

The results obtained by the proposed approach are shown in Table 5; in this case, 1R is the only meta-classifier for which a best result is not obtained in any dataset. Best SG meta-classifier is also different in 10 of the 20 classification problems, which indicates different structures between the standard SG and the CSS approaches. Regarding the best results obtained, RF appears as meta-classifier in 10 datasets, while KNN, NB and SVM obtain as meta the best results in 5 classification problems.

A graphical image of the results obtained by the CSS paradigm is shown in Fig. 6. Comparing them with the results of standard SG (Fig. 5), it can be seen that, together with the improvement of accuracy, the variance of results for each database is reduced. In Table 6 means and standard deviations of accuracy values are listed for different types of classifiers and for each database. The best means and lowest standard deviation have been marked. As it can be seen. the proposed approach obtains the best mean in all the databases and the lowest standard deviation in most of the databases. We consider that this fact shows the effectiveness of our method and that it is not related with the type of classifier that is used as meta-classifier.

In Figs. 7 and 8 comparisons of CSS, SG and single classifiers are shown for each database. These figures show the best results obtained for each base classifier (single) and using this classifier type as meta (SG, CSS).

To emphasize the differences between paradigms, Fig. 9 shows a comparison of best results (maximum accuracy) obtained for each database. As can be seen, CSS paradigm outperforms the others in all used datasets except breast-cancer. It is worth noting that the aim of this paper is to present a new competitive approach; it is not the authors' intention to show a MSC that always sets the best results, although this has been obtained with the selected datasets.

Finally, to have a more accurate idea of the kind of classifier subsets selected by CSS method, Table 7 shows the configurations that give the best results for each dataset.

4.5.5. *Statistical tests*

According to [65], we have used the Iman–Davenport test to detect statistical differences among the different strategies. This test rejects the null hypothesis of equivalence between algorithms, since p-value ($2.2e-16$) is lower than our α-value (0.1). Thus, we have applied Shaffer post hoc test in order to find out which algorithms

Table 8
Significant differences obtained with Shaffer post hoc test.

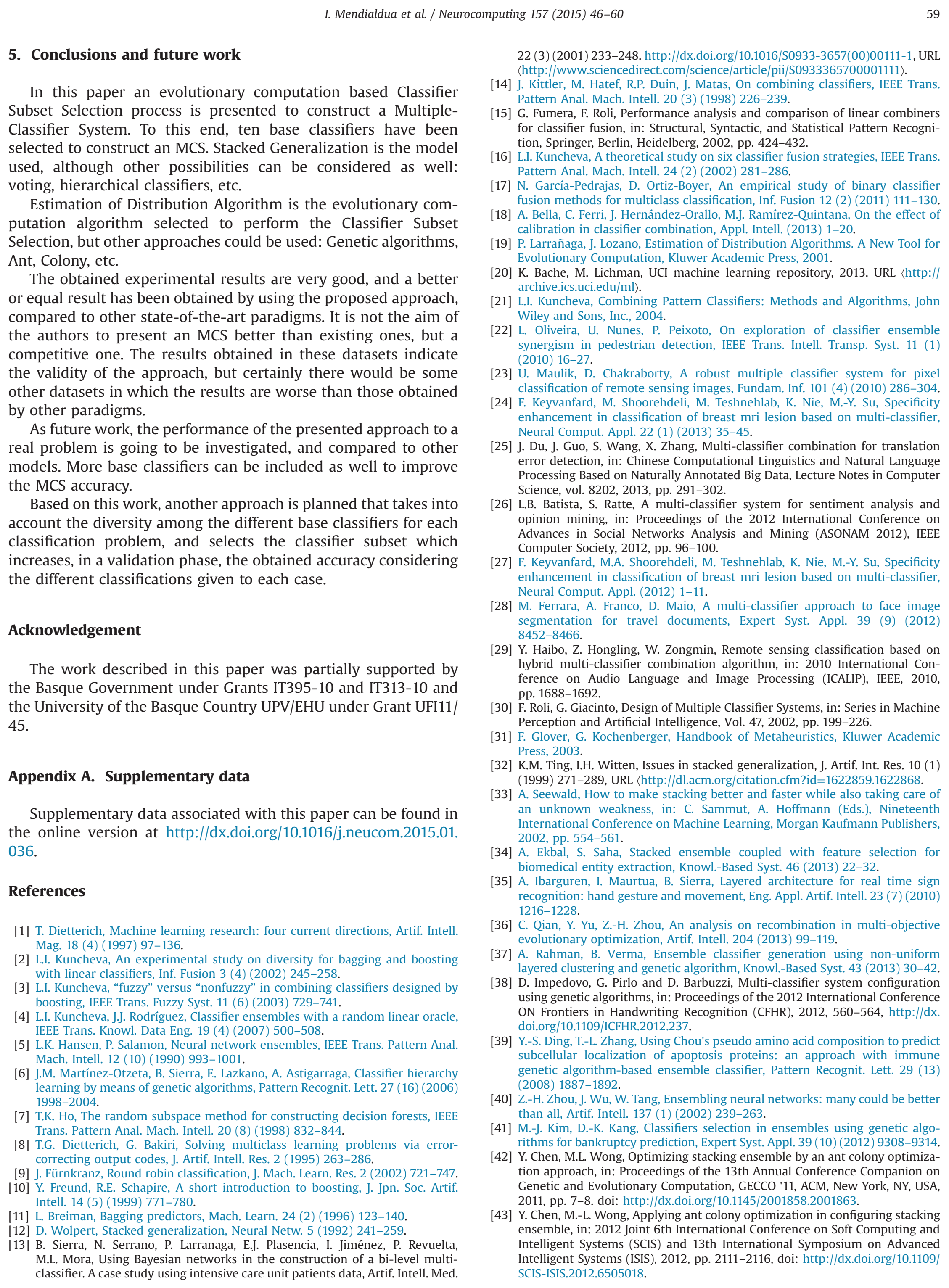
CSS	Individual				Ensemble			Stacked Generalization				
	Win		Draw	Loose	Win	Draw	Loose	Win		Draw		Loose
RF	ALL		–	–	BAG-C4.5	STC	–	1R	KNN			–
					BAG-REP			RIP	NB	RF		
					BOS-DS			C4.5	K*	SVM		
					BOS-C4.5			BN	NBT			
SVM	ALL		–	–	BAG-C4.5	STC	–	1R	KNN			–
					BAG-REP			RIP	NB	RF		
					BOS-DS			C4.5	K*	SVM		
					BOS-C4.5			BN	NBT			
K*	ALL		–	–	BAG-C4.5	STC	–	1R	KNN	BN		–
					BAG-REP			RIP	NB	RF		
					BOS-DS			C4.5	K*	SVM		
					BOS-C4.5			NBT				
C4.5	ALL		–	–	BAG-C4.5	STC	–	1R	KNN	BN		–
					BAG-REP			RIP	NB	RF		
					BOS-DS			C4.5	K*	SVM		
					BOS-C4.5			NBT				
RIP	ALL		–	–	BAG-REP		–	1R	KNN	BN		–
					BOS-DS	STC		RIP	NB	RF		
					BOS-C4.5	BAG-C4.5		C4.5	K*	SVM		
								NBT				
KNN	1R	KNN	SVM	–			–	1R	KNN	NB		–
	RIP	NB			BAG-REP	STC		RIP		BN		
	C4.5	K*			BOS-DS	BAG-C4.5		C4.5		RF		
	BN	NBT			BOS-C4.5			K*		SVM		
	RF							NBT				
NB	1R	KNN	SVM	–			–	1R	KNN	NB		–
	RIP	NB			BAG-REP	STC		RIP		BN		
	C4.5	K*			BOS-DS	BAG-C4.5		C4.5		RF		
	BN	NBT			BOS-C4.5			K*		SVM		
	RF							NBT				
NBT	1R	KNN		–	BAG-REP		–	1R	KNN	NB		–
	RIP	NB	C4.5		BOS-DS	STC		RIP	C4.5	BN		
	K*	BN	SVM		BOS-C4.5	BAG-C4.5		K*		RF		
	NBT	RF						NBT		SVM		
BN	1R	KNN		–	BAG-REP		–	1R		RIP	NB	–
	RIP	NB	C4.5		BOS-DS	STC		KNN		C4.5	BN	
	K*	BN	SVM		BOS-C4.5	BAG-C4.5		K*		RF		
	NBT	RF						NBT		SVM		
1R	–		ALL	–	–	ALL	–	–		ALL		–

Table 9
The *p*-values obtained with Wilcoxon test.

CSS	SG-RF	SG-SVM	STC
1R	−(0.01923)	=(0.10540)	−(0.02148)
KNN	+(0.00032)	+(0.00365)	+(0.00584)
RIPPER	+(0.00001)	+(0.00058)	+(0.00058)
NB	+(0.00401)	+(0.00199)	+(0.00745)
C4.5	+(0.00027)	+(0.00068)	+(0.00008)
K*	+(3.8E-006)	+(0.00010)	+(0.00008)
BN	+(0.00032)	+(0.00315)	+(0.00745)
NBT	+(0.00233)	+(0.00121)	+(0.02944)
RF	+(0.00014)	+(0.00050)	+(0.00005)
SVM	+(0.00004)	+(1.9E-006)	+(1.9E-006)

are distinctive among them. Table 8 shows the statistical differences that our approach has obtained with the rest of methods. The table shows that when RF or SVM are used as meta-classifier in CSS (CSS-RF and CSS-SVM) best results are obtained, closely followed by CSS-K* and CSS-C4.5. In the case of CSS-RF and CSS-SVM, they significantly improve most of the cases and draw with only 3 of them: StackingC, SG-RF and SG-SVM. On the other hand, CSS-1R obtains the worst results because it draws with all the cases. We also want to emphasize that our algorithm never gets worse results.

In Table 8 it can be seen that CSS does not significantly improve StackingC, SG-RF and SG-SVM with any of the meta-classifiers. Because of that we have compared these 3 methods with our approach applying Wilcoxon signed-rank test by pairwise. Table 9 shows the *p*-values obtained, where "+" symbol implies that the CSS is statistically better than the confronting one, whereas "=" means that there are no significant differences between the compared algorithms and "−" means that the CSS is statistically worse. It can be seen that, with the exception of 1R, the rest of meta-classifiers significantly improve the rest of algorithms. Viewing the results of both statistical tests, considerably demonstrates the strength of our new method.



5. Conclusions and future work

In this paper an evolutionary computation based Classifier Subset Selection process is presented to construct a Multiple-Classifier System. To this end, ten base classifiers have been selected to construct an MCS. Stacked Generalization is the model used, although other possibilities can be considered as well: voting, hierarchical classifiers, etc.

Estimation of Distribution Algorithm is the evolutionary computation algorithm selected to perform the Classifier Subset Selection, but other approaches could be used: Genetic algorithms, Ant, Colony, etc.

The obtained experimental results are very good, and a better or equal result has been obtained by using the proposed approach, compared to other state-of-the-art paradigms. It is not the aim of the authors to present an MCS better than existing ones, but a competitive one. The results obtained in these datasets indicate the validity of the approach, but certainly there would be some other datasets in which the results are worse than those obtained by other paradigms.

As future work, the performance of the presented approach to a real problem is going to be investigated, and compared to other models. More base classifiers can be included as well to improve the MCS accuracy.

Based on this work, another approach is planned that takes into account the diversity among the different base classifiers for each classification problem, and selects the classifier subset which increases, in a validation phase, the obtained accuracy considering the different classifications given to each case.

Acknowledgement

The work described in this paper was partially supported by the Basque Government under Grants IT395-10 and IT313-10 and the University of the Basque Country UPV/EHU under Grant UFI11/45.

Appendix A. Supplementary data

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.neucom.2015.01.036.

References

[1] T. Dietterich, Machine learning research: four current directions, Artif. Intell. Mag. 18 (4) (1997) 97–136.
[2] L.I. Kuncheva, An experimental study on diversity for bagging and boosting with linear classifiers, Inf. Fusion 3 (4) (2002) 245–258.
[3] L.I. Kuncheva, “fuzzy” versus “nonfuzzy” in combining classifiers designed by boosting, IEEE Trans. Fuzzy Syst. 11 (6) (2003) 729–741.
[4] L.I. Kuncheva, J.J. Rodríguez, Classifier ensembles with a random linear oracle, IEEE Trans. Knowl. Data Eng. 19 (4) (2007) 500–508.
[5] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990) 993–1001.
[6] J.M. Martínez-Otzeta, B. Sierra, E. Lazkano, A. Astigarraga, Classifier hierarchy learning by means of genetic algorithms, Pattern Recognit. Lett. 27 (16) (2006) 1998–2004.
[7] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844.
[8] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Intell. Res. 2 (1995) 263–286.
[9] J. Fürnkranz, Round robin classification, J. Mach. Learn. Res. 2 (2002) 721–747.
[10] Y. Freund, R.E. Schapire, A short introduction to boosting, J. Jpn. Soc. Artif. Intell. 14 (5) (1999) 771–780.
[11] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[12] D. Wolpert, Stacked generalization, Neural Netw. 5 (1992) 241–259.
[13] B. Sierra, N. Serrano, P. Larranaga, E.J. Plasencia, I. Jiménez, P. Revuelta, M.L. Mora, Using Bayesian networks in the construction of a bi-level multi-classifier. A case study using intensive care unit patients data, Artif. Intell. Med. 22 (3) (2001) 233–248. http://dx.doi.org/10.1016/S0933-3657(00)00111-1, URL ⟨http://www.sciencedirect.com/science/article/pii/S0933365700001111⟩.
[14] J. Kittler, M. Hatef, R.P. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998) 226–239.
[15] G. Fumera, F. Roli, Performance analysis and comparison of linear combiners for classifier fusion, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer, Berlin, Heidelberg, 2002, pp. 424–432.
[16] L.I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2) (2002) 281–286.
[17] N. García-Pedrajas, D. Ortiz-Boyer, An empirical study of binary classifier fusion methods for multiclass classification, Inf. Fusion 12 (2) (2011) 111–130.
[18] A. Bella, C. Ferri, J. Hernández-Orallo, M.J. Ramírez-Quintana, On the effect of calibration in classifier combination, Appl. Intell. (2013) 1–20.
[19] P. Larrañaga, J. Lozano, Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation, Kluwer Academic Press, 2001.
[20] K. Bache, M. Lichman, UCI machine learning repository, 2013. URL ⟨http://archive.ics.uci.edu/ml⟩.
[21] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley and Sons, Inc., 2004.
[22] L. Oliveira, U. Nunes, P. Peixoto, On exploration of classifier ensemble synergism in pedestrian detection, IEEE Trans. Intell. Transp. Syst. 11 (1) (2010) 16–27.
[23] U. Maulik, D. Chakraborty, A robust multiple classifier system for pixel classification of remote sensing images, Fundam. Inf. 101 (4) (2010) 286–304.
[24] F. Keyvanfard, M. Shoorehdeli, M. Teshnehlab, K. Nie, M.-Y. Su, Specificity enhancement in classification of breast mri lesion based on multi-classifier, Neural Comput. Appl. 22 (1) (2013) 35–45.
[25] J. Du, J. Guo, S. Wang, X. Zhang, Multi-classifier combination for translation error detection, in: Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data, Lecture Notes in Computer Science, vol. 8202, 2013, pp. 291–302.
[26] L.B. Batista, S. Ratte, A multi-classifier system for sentiment analysis and opinion mining, in: Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), IEEE Computer Society, 2012, pp. 96–100.
[27] F. Keyvanfard, M.A. Shoorehdeli, M. Teshnehlab, K. Nie, M.-Y. Su, Specificity enhancement in classification of breast mri lesion based on multi-classifier, Neural Comput. Appl. (2012) 1–11.
[28] M. Ferrara, A. Franco, D. Maio, A multi-classifier approach to face image segmentation for travel documents, Expert Syst. Appl. 39 (9) (2012) 8452–8466.
[29] Y. Haibo, Z. Hongling, W. Zongmin, Remote sensing classification based on hybrid multi-classifier combination algorithm, in: 2010 International Conference on Audio Language and Image Processing (ICALIP), IEEE, 2010, pp. 1688–1692.
[30] F. Roli, G. Giacinto, Design of Multiple Classifier Systems, in: Series in Machine Perception and Artificial Intelligence, Vol. 47, 2002, pp. 199–226.
[31] F. Glover, G. Kochenberger, Handbook of Metaheuristics, Kluwer Academic Press, 2003.
[32] K.M. Ting, I.H. Witten, Issues in stacked generalization, J. Artif. Int. Res. 10 (1) (1999) 271–289, URL ⟨http://dl.acm.org/citation.cfm?id=1622859.1622868.
[33] A. Seewald, How to make stacking better and faster while also taking care of an unknown weakness, in: C. Sammut, A. Hoffmann (Eds.), Nineteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, 2002, pp. 554–561.
[34] A. Ekbal, S. Saha, Stacked ensemble coupled with feature selection for biomedical entity extraction, Knowl.-Based Syst. 46 (2013) 22–32.
[35] A. Ibarguren, I. Maurtua, B. Sierra, Layered architecture for real time sign recognition: hand gesture and movement, Eng. Appl. Artif. Intell. 23 (7) (2010) 1216–1228.
[36] C. Qian, Y. Yu, Z.-H. Zhou, An analysis on recombination in multi-objective evolutionary optimization, Artif. Intell. 204 (2013) 99–119.
[37] A. Rahman, B. Verma, Ensemble classifier generation using non-uniform layered clustering and genetic algorithm, Knowl.-Based Syst. 43 (2013) 30–42.
[38] D. Impedovo, G. Pirlo and D. Barbuzzi, Multi-classifier system configuration using genetic algorithms, in: Proceedings of the 2012 International Conference ON Frontiers in Handwriting Recognition (CFHR), 2012, 560–564, http://dx.doi.org/10.1109/ICFHR.2012.237.
[39] Y.-S. Ding, T.-L. Zhang, Using Chou's pseudo amino acid composition to predict subcellular localization of apoptosis proteins: an approach with immune genetic algorithm-based ensemble classifier, Pattern Recognit. Lett. 29 (13) (2008) 1887–1892.
[40] Z.-H. Zhou, J. Wu, W. Tang, Ensembling neural networks: many could be better than all, Artif. Intell. 137 (1) (2002) 239–263.
[41] M.-J. Kim, D.-K. Kang, Classifiers selection in ensembles using genetic algorithms for bankruptcy prediction, Expert Syst. Appl. 39 (10) (2012) 9308–9314.
[42] Y. Chen, M.L. Wong, Optimizing stacking ensemble by an ant colony optimization approach, in: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 7–8. doi: http://dx.doi.org/10.1145/2001858.2001863.
[43] Y. Chen, M.-L. Wong, Applying ant colony optimization in configuring stacking ensemble, in: 2012 Joint 6th International Conference on Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012, pp. 2111–2116, doi: http://dx.doi.org/10.1109/SCIS-ISIS.2012.6505018.

[44] P. Shunmugapriya, S. Kanmani, Optimization of stacking ensemble configurations through artificial bee colony algorithm, Swarm Evol. Comput. 12 (2013) 24–32.
[45] A. Ledezma, R. Aler, A. Sanchís, D. Borrajo, Ga-stacking: evolutionary stacked generalization, Intell. Data Anal. 14 (1) (2010) 89–119.
[46] B. Sierra, N. Serrano, P. Larranaga, E.J. Plasencia, I. Inza, J.J. Jiménez, P. Revuelta, M.L. Mora, Using Bayesian networks in the construction of a bi-level multi-classifier. A case study using intensive care unit patients data, Artif. Intell. Med. 22 (3) (2001) 233–248.
[47] D.W. Aha, R.L. Bankert, Feature selection for case-based classification of cloud types: an empirical comparison, in: Proceedings of the AAAI'94 Workshop on Case-Based Reasoning, 1994, pp. 106–112.
[48] I. Inza, P. Larrañaga, R. Etxeberria, B. Sierra, Feature Subset Selection by Bayesian network-based optimization, Artif. Intell. 123 (1–2) (2000) 157–184.
[49] R. Etxeberria, P. Larranaga, Global optimization using Bayesian networks, in: Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99), 1999, pp. 332–339.
[50] I. Inza, P. Larrañaga, B. Sierra, Feature subset selection by Bayesian networks: a comparison with genetic and sequential algorithms, Int. J. Approx. Reason. 27 (2) (2001) 143–164.
[51] C. Echegoyen, A. Mendiburu, R. Santana, J.A. Lozano, Toward understanding edas based on Bayesian networks through a quantitative analysis, IEEE Trans. Evol. Comput. 16 (2) (2012) 173–189.
[52] M. Hall, E. Frank, G. Holmes, B.P.P., Reutemann, I. Witten, The weka data mining software: an update, SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[53] R. Holte, Very simple classification rules perform well on most commonly used datasets, Mach. Learn. 11 (1993) 63–91.
[54] D. Aha, D. Kibler, M. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1991) 37–66.
[55] W. Cohen, Fast effective rule induction, in: 12th International Conference on Machine Learning, Morgan Kaufmann Publishers, 1995, pp. 115–123.
[56] B. Cestnik, Estimating probabilities: a crucial task in machine learning, in: Proceedings of the European Conference on Artificial Intelligence, 1990, pp. 147–149.
[57] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.
[58] J. Cleary, L. Trigg, K*: an instance-based learner using an entropic distance measure, in: 12th International Conference on Machine Learning, 1995, pp. 108–114.
[59] B. Sierra, E. Lazkano, E. Jauregi, I. Irigoien, Histogram distance-based Bayesian network structure learning: a supervised classification specific approach, Decision Support Syst. 48 (1) (2009) 180–190.
[60] R. Kohavi, Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid, in: Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 202–207.
[61] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[62] D. Meyer, F. Leisch, K. Hortnik, The support vector machine under test, Neurocomputing 55 (1) (2003) 169–186.
[63] M. Stone, Cross-validation choice and assessment of statistical procedures, J. R. Stat. Soc. 36 (1974) 111–147.
[64] W. Buntine, Theory Refinement on Bayesian Networks, Morgan Kaufmann Publishers (1991) 52–60.
[65] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.

Andoni Arruti is an Associate Professor in the Department of Computer Architecture and Technology at the University of the Basque Country. He received his M.Sc. degree in physics from the University of Navarra, in Spain, in 1985. He received the Ph.D. in electronics from the University of Navarra, in Spain, in 1990. Dr Arruti is presently a researcher in the fields of Biosignals Processing and Machine Learning, and he is working in the Laboratory of Human–Computer interaction for Special Needs. This research group is located in the Department of Computer Architecture and Technology, Computer Science Faculty, at Donostia-San Sebastián.

Iñigo Mendialdua received the B.Sc. degree in Computer Science from the University of the Basque Country, Spain, in 2007 and is pursuing the Ph.D. degree from the Department of Computer Science and Artificial Intelligence, University of the Basque Country. His current research interests include distance-based algorithms, multi-classifiers and decomposition strategies.

Ekaitz Jauregi is a posdoc fellow in the Department of Computer Science and Artificial Intelligence of the University of the Basque Country. He is a member of the Robotics and Autonomous Systems Group. His research interests include Machine Learning, Behaviour Based Architectures and Robotics.

Elena Lazkano is an Assistant Professor in the Computer Sciences and Artificial Intelligence Department at the University of the Basque Country (UPV/EHU). She received her B.Sc. in Computer Sciences in 1992 (UPV/EHU), M.Sc. in Artificial Intelligence (Katholieke Universiteit Leuven, Belgium, 1993) and Ph.D. in Computer Sciences in 2004 (UPV/EHU). She is codirector of the Robotics and Autonomous Systems Group in Donostia-San Sebastian. Dr. Lazkano is presently a researcher in the fields of Robotics, being a member of the Program Committee of International Conferences and reviewer of International Journals; she is working on the development of a behaviour-based distributed probabilistic control architecture.

Basilio Sierra is a Full Professor in the Computer Sciences and Artificial Intelligence Department at the University of the Basque Country. He received his B.Sc. in Computer Sciences in 1990, M.Sc. in Computer Science and Architecture in 1992 and Ph.D. in Computer Sciences in 2000 at the University of the Basque Country. He is codirector of the Robotics and Autonomous Systems Group in Donostia-San Sebastian. Prof. Sierra is presently a researcher in the fields of Robotics and Machine Learning, and he is working on the use of different paradigms to improve behaviours.

Part IV

Articles Related to Class Binarization

13

$New\ One\ Versus^{All}_{One}$ method: NOV@

Title: $New\ One\ Versus^{All}_{One}$ method: NOV@

Authors: A. Arruti, I. Mendialdua, B. Sierra, E. Lazkano and E. Jauregi.

Journal: Expert Systems with Applications

Publisher: Elsevier

Year: 2014

DOI: http://dx.doi.org/10.1016/j.eswa.2014.04.010

Expert Systems with Applications 41 (2014) 6251–6260

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

New One Versus$^{All}_{One}$ method: NOV@

A. Arruti [a], I. Mendialdua [b,*], B. Sierra [b], E. Lazkano [b], E. Jauregi [b]

[a] *Department of Computer Architecture and Technology, University of the Basque Country UPV/EHU, Donostia-San Sebastian 20018, Spain*
[b] *Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, Donostia-San Sebastian 20018, Spain*

A R T I C L E I N F O

Keywords:
Decomposition strategies
One against One
One against All

A B S T R A C T

Binarization strategies decompose the original multi-class dataset into multiple two-class subsets, learning a different binary model for each new subset. One-vs-All (OVA) and One-vs-One (OVO) are two of the most well-known techniques: One-vs-One separates a pair of classes in each binary sub-problem, ignoring the remaining ones; and One-vs-All distinguishes one class from all the other classes. In this paper, we present two new OVA and OVO combinations where the best base classifier is applied in each sub-problem. The first method is called OVA + OVO since it combines the outputs obtained by OVA and OVO decomposition strategies. The second combination is named *New One Versus$^{All}_{One}$* (NOV@), and its objective is to solve the problems found in OVA when different base classifiers are used in each sub-problem. In order to validate the performance of the new proposal, an empirical study has been carried out where the two new methods are compared with other well-known decomposition strategies from the literature. Experimental results show that both methods obtain promising results, especially NOV@.

1. Introduction

The goal in a supervised classification problem consists in classifying a new unlabeled example x in its correct class using a training set. Let $TR = \{x_i, \theta_i\}_{i=1}^N$ denote a training set of N well-labeled examples, where x_i represents the ith individual feature vector and θ_i represents the class the individual belongs to. In the particular case of the K-class problem, being $\theta \in \{1, \ldots, K\}$, the class label is commonly defined as an integer. Based on the *TR*, the supervised classification techniques create a "general rule" or a function which is used to classify each new unlabeled case. This general rule is also known as a classifier.

For some types of classifiers, such as SVM, it is much more easier to build a classifier to distinguish just between two classes. However, many real world problems are multi-class problems, i.e. $K > 2$. In view of that, there are some techniques, known as class-binarization techniques, which divide the original multi-class problem into many binary classification problems. These techniques are two-step methods: in the first step, called decomposition step, a classifier is learned for each of the sub-problems, and in the second step, called combination step, the outputs of these classifiers are combined to obtain the final prediction.

There are three different class-binarization strategies: "One-vs-All" (OVA), "One-vs-One" (OVO) and "Error Correcting Output Codes" (ECOC). OVA and OVO are the most relevant ones in the literature due to their simplicity and clarity.

- One-vs-All (OVA) (Anand, Mehrotra, Mohan, & Ranka, 1995): In each sub-problem one class is compared with the rest of classes.
- One-vs-One (OVO) (Fürnkranz, 2002): In each sub-problem only the cases belonging to two classes are compared with each other, and the remaining ones are ignored.
- Error Correcting Output Codes (ECOC) (Dietterich & Bakiri, 1995): In each sub-problem all the classes are grouped into two groups, and the two groups are compared with each other.

The procedure followed in the classical binary classification strategies is to use the same base classifiers in each binary sub-problem. However, if the selected base classifier does not correctly discriminate in some of the sub-problems, the obtained results will be wrong. To overcome this drawback, each sub-problem can be treated as an independent classification problem so that a different base classifier can be used for each sub-problem.

In this paper, we propose two new combinations of the methods OVA and OVO. We compare these two methods with other class-binarization strategies over 20 UCI databases, and obtain promising results. In all methods we try to find the best base classifiers for each sub-problem. To do so, we have chosen several well-known classifiers from different Machine Learning

* Corresponding author. Address: Robotics and Autonomous System Research Group, University of the Basque Country UPV/EHU, Donostia-San Sebastian 20018, Spain. Tel.: +34 943 015107.
E-mail address: inigo.mendialdua@ehu.es (I. Mendialdua).

http://dx.doi.org/10.1016/j.eswa.2014.04.010
0957-4174/© 2014 Elsevier Ltd. All rights reserved.

paradigms: SVM, C4.5 Decision Tree, Ripper, K-NN, Multilayer Perceptron and Naive Bayes. The first combination that we have proposed is called OVA + OVO. OVA + OVO basically combines the sub-problems obtained after the OVA and OVO methods are applied. Comparing this new method with other class-binarization strategies we have found that strategies – such as OVA – are not suitable when different base classifiers are used in each sub-problem. Therefore, we propose a second approach that we have called *New One Versus*$^{All}_{One}$ (NOV@). NOV@ is an extension of OVA: at decomposition time OVA is applied, whereas at combination time the majority vote is used to make the final decision. In case of a tie among several classes, OVO is applied for tie-breaking – taking just into account the tied classes.

Although in the specialized literature, there are several proposals that select the best base classifier for each sub-problem, none of them compares different class-binarization techniques in order to study how each technique performs. In our work we have compared the two new methods with other state-of-the-art class-binarization strategies in an empirical study.

The rest of the paper is organized as follows. In Section 2 we review the decomposition techniques, with special attention to OVA and OVO strategies. In Section 3 we show the compatibility between OVA and OVO strategies and we present the first proposed method: OVA + OVO. Section 4 describes our second approach and Section 5 shows the results of the experiments carried out. Finally, Section 6 states the conclusions of our work and suggests future research lines.

2. Class-binarization

The first class-binarization strategies were made to solve the problems that some base classifiers have for the multi-class problems, since algorithms such as Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP), worked better for binary problems. However, due to the good results obtained, the use of these strategies has been extended to other base classifiers, such as Ripper (Fürnkranz, 2002) or C4.5 (Polat & Güneş, 2009).

Class-binarization is composed of two steps: decomposition and combination.

In the decomposition step the original problem is divided into several binary sub-problems. The most popular strategies consist of grouping classes, in this way each binary classifier compares two groups of classes between them. Commonly the code-matrix is used to represent how the classes are grouped.

Fig. 1 shows a code-matrix example, where 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. In Fig. 1 how a 5-class problem $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$ is decomposed into 5 binary problems $\{f_1, f_2, f_3, f_4, f_5\}$ can be seen. 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 in 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.

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

Fig. 1. Example of a code-matrix.

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

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

2.1. One-vs-All (OVA)

OVA decomposition scheme divides a K class multi-class problem, $\theta_1, \ldots, \theta_K$, into K two-class problems, where each binary problem discriminates one class from the others.

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

As one class is compared with all the other classes, most of the binary sub-problems are unbalanced. It is known that one of the drawbacks of an unbalanced problem is the underestimation for the minority classes, thereby the most represented class is selected in most cases. In view of that, in OVA it is very common that all sub-problems return a class-negative prediction, hence, ties are usual in the final decision when the majority vote is used. Thus, it is more efficient to use the confidence level of each classifier to decide the final output. The class with the highest confidence is the selected decision.

2.2. One-vs-One (OVO)

OVO decomposition scheme divides a K class multi-class problem, $\theta_1, \ldots, \theta_K$, into $K(K-1)/2$ two-class sub-problems. In each sub-problem a classifier is learned 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.

In Fig. 2(b) OVO's code-matrix for 4 classes can be observed: in each classifier one class is represented as +1 class, another one is represented as -1 and the remaining 2 classes are represented as 0.

There are several strategies to combine the output, the simplest way to combine the outputs is to use the majority vote strategy (Friedman, 1996; Fürnkranz, 2002) also called as Max-Wins; the most voted class is the selected one. An immediate extension is the Weighted Voting (WV): to use the confidence level of each binary problem as a vote. Its robustness has been shown in Galar, Fernández, Barrenechea, Bustince, and Herrera (2011). Hastie and Tibshirani (1998) proposed a new method called Pairwise Coupling (PC). The aim of the method is to find the best approximation of the class posterior probabilities given the posterior probabilities of the pairwise sub-problems. To do so, they transform the problem into an iterative problem where they try to minimize the average weighted Kullback–Leiber divergence between the obtained pairwise estimates and the true pairwise probability values.

OVO has several drawbacks 3 of the main disadvantages of OVO are the following:

1. Unclassifiable regions: It is possible that each binary classifier votes for a different class, hence there is no winner. Thus, some tie-breaking technique has to be applied.
2. Number of classifiers: Compared with OVA, it can be seen that OVO creates more sub-problems. Moreover, the disadvantage of having so many sub-problems is that most of them are

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

Fig. 2. OVA and OVO code-matrix.

irrelevant and they are forced to give wrong answers for many instances, because each classifier must assign every pattern to one of two classes. If a pattern belongs to class *i*, all the classifiers that are not trained to differentiate this class will cast wrong votes. However, OVO uses fewer examples in each sub-problem and, thus, has more freedom for fitting a decision boundary between the two classes.

3. Weak classifiers: The classical way selects the optimal base classifier for the database and all the sub-problems are classified with this classifier. As there are too many sub-problems, it is possible that this base classifier has difficulties to distinguish between all of them, thus, the classifiers return wrong results. This raises the question – should the same base classifier be used on all sub-problems or should sub-problems be tuned independently?

Several proposals has been developed in the literature in order to solve these problems.

In order to resolve the unclassifiable regions Platt, Cristianini, and Shawe-Taylor (2000) published a new combination proposal called Decision Directed Acyclic Graph (DDAG). DDAG builds a rooted binary acyclic graph where in each node a classifier discriminates between two classes. The final answer is the class assigned by the leaf node. Liu, Hao, and Yang (2007) proposed a tie-breaking technique, where OVO is applied using only the examples in the unclassifiable region.

On the other hand, other approaches try to reduce the number of binary classifiers in OVO using the Dynamic Classifier Selection (Bagheri, Gao, & Escalera, 2012; Galar, Fernández, Barrenechea, Bustince, & Herrera, 2013). Other authors propose the use of hierarchical structure. Fei and Liu (2006) proposed a new architecture called Binary Tree of SVM (BTS). BTS is a binary tree where in each node two classes are distinguished. The main idea of BTS is to use the separating plane for these two classes, also to distinguish other classes. Chen, Wang, and Wang (2009) proposed a new BTS version where they tried to select the binary SVM with the fewest number of separating lines. Following the same idea, to reduce the number of classifiers, the hierarchical structure has been extended to other class-binarization strategies (Ghaffari & Yazdi, 2013; Lorena & de Carvalho, 2010; Madzarov & Gjorgjevikj, 2009; Pujol, Radeva, & Vitria, 2006).

Finally other approaches try to solve the weak classifier problem using each sub-problem independently. Szepannek, Bischl, and Weihs (2009) proposed to extend OVO selecting the optimal classifier for each pair of classes, i.e. the base classifier which obtains the best result. Something similar was proposed by Lebrun, Lezoray, Charrier, and Cardot (2007), where they tried to find the best hyper-parameters of the SVM for each sub-problem. Due to the high number of hyper-parameters in the SVM, they proposed to use an evolutionary algorithm. The experimental results of both works showed that they outperformed the classical individual base-classifier option. Liepert (2003) also proposed a similar approach to Lebrun, but she concluded that the selection of the best models for each binary-classifier does not obtain a significant improvement.

In his PhD thesis, Reid (2010) concluded that despite it is better to use the same base classifier for all the sub-problems when decision boundaries of the sub-problems have similar shapes, in the case where the decision boundaries have a different shape it is better to treat sub-problems independently.

2.3. Related work

In several works in the literature, OVA and OVO have been compared, showing that in most of the cases OVO outperformed OVA (Fürnkranz, 2002; Hsu & Lin, 2002). Rifkin and Klautau (2004) did not consider that these experiments were carefully controlled and they demonstrated that when OVA classifier is well-tuned it performs as well as OVO.

In some recent works, it is possible to find some reviews related with the class-binarization strategies (Lorena, de Carvalho, & Gama, 2008). Other recent works made empirical studies for different binarization strategies (Galar et al., 2011; García-Pedrajas & Ortiz-Boyer, 2011). Both works analyse the behavior of OVA and OVO for different base classifiers and the results of both works concluded that OVO outperformed OVA. However, neither work dares to contradict Rifkin and Klautau (2004) arguing that they did not use fine-tuned classifiers.

Some authors proposed new approaches based on the combination of OVA and OVO. On the one hand, Moreira and Mayoraz (1998) proposed to apply OVO taking into account the probability that the new example belonged to each pair of classes. This probability was obtained following the OVA idea: creating a classifier that distinguishes between the two classes and the rest of the classes. On the other hand, García-Pedrajas and Ortiz-Boyer (2006) and Ko and Byun (2003) proposed a very similar idea to combine OVA and OVO. In an interesting motivation section García-Pedrajas and Ortiz-Boyer (2006) showed that in the majority of the cases the correct class was between the two largest confident outputs of OVA. Thus, in their new method they obtain the two classes with the highest confidence level in OVA first. After that, OVO is applied and a classifier is built only taking into account the cases that belong to these two classes. This method was called All-And-One (A&O) (García-Pedrajas & Ortiz-Boyer, 2006).

3. Combining OVA and OVO

Hansen and Salamon (1990) showed that in order to obtain good performance when classifiers are combined, the classifiers should be diverse among them. It is said that two algorithms are diverse when they commit different errors. If two algorithms are not diverse they commit the same errors, whereas if they are diverse they may be able to correct the committed errors.

It must be said that, although several diversity measures exists in the literature Kuncheva and Whitaker (2003), they have been proved to be ineffective Tang, Suganthan, and Yao (2006). Hence, instead of applying those measures, we choose to perform a simple experiment using decision trees as base classifier, in order to conclude if OVA and OVO are compatible.

Twenty databases are used to test our hypothesis. All of them are obtained from the *UCI Machine Learning Repository* (Bache & Lichman, 2013). The characteristics of the databases are given in Table 1.

In Table 2 we show the results of the compatibility test. In the first two columns we show the accuracy obtained by OVO and OVA, while in the third the accuracy of OVA is shown considering only the cases where OVO makes errors. In the fourth column the accuracy of OVO is shown for the cases that OVA has failed.

It is interesting to note that in the case where one strategy fails the other gets a hit rate higher than 20% in most of the cases, which leads us to consider that the methods are compatible and that they could correct some of the errors made by the other strategy.

3.1. Our first proposal to combine OVA and OVO

This new method combines the sub-problems obtained applying OVA and OVO decomposition strategies. To take the final decision the results obtained for each sub-problem are combined applying the majority vote. We have called this new method OVA + OVO. Following the idea proposed by Szepannek et al. (2009), we apply the most reliable base classifier for each sub-problem.

Table 1
The characteristics of the 20 databases used in this experiment.

Domain	Num. of instances	Num. of attributes	Num. of classes
Abalone	4177	8	29
Annealing	798	38	5
Arrhythmia	452	279	13
Balance	625	4	3
Car	1728	6	4
Cmc	1473	9	3
Dermatology	366	33	6
Ecoli	336	7	8
Flare	1389	11	6
Glass	214	9	6
Iris	150	4	3
Nursery	12960	8	5
Page-blocks	5473	10	5
Optdigits	5620	64	10
Pendigits	10992	16	10
Satimage	6435	36	7
Segment	2310	19	7
Vehicle	846	18	4
Waveform	5000	21	3
Wine	178	13	3
Winequality red	1599	10	6
Winequality white	4898	10	7
Yeast	1484	8	10
Zoo	101	16	7

3.1.1. Decomposition

In our method, we propose to combine the sub-problems obtained with OVA and OVO; on the one hand, we create several sub-problems where the classes are compared by pairs, ignoring the other classes, and on the other hand, we create several sub-problems which compare one class with all the other classes.

Fig. 3 shows the code-matrix of the sub-problems obtained in the decomposition phase in a 4-class problem. The 4 columns of the left are the sub-problems obtained with OVA and the next 6 columns are the sub-problems obtained with OVO.

We consider that the combination of these methods could obtain a classifier that outperforms both methods separately for two reasons:

$$\left(\begin{array}{cccccccccc} \multicolumn{4}{c}{\overbrace{\hspace{8em}}^{One-Vs-All}} & \multicolumn{6}{c}{\overbrace{\hspace{12em}}^{One-Vs-One}} \\ +1 & -1 & -1 & -1 & +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & +1 & -1 & -1 & -1 & 0 & 0 & +1 & +1 & 0 \\ -1 & -1 & +1 & -1 & 0 & -1 & 0 & -1 & 0 & +1 \\ -1 & -1 & -1 & +1 & 0 & 0 & -1 & 0 & -1 & -1 \end{array}\right)$$

Fig. 3. Code-matrix of OVA + OVO.

1. As shown in Section 3 both methods are able to correct the errors committed by the other one.
2. As mentioned before, when majority vote is used for the final decision it is common for both, OVA and OVO, to produce ties.
   - In the case of OVA, it is possible that all sub-problems return a negative-class prediction.
   - In the case of OVO, the most voted class could be more than one.

Therefore, we consider that combining their outputs could serve to break some of these ties.

3.1.2. Best base classifier for each sub-problem

Each sub-problem is treated independently from the others, the optimal base classifier for each one is sought. In a validation process each binary sub-problem is tested with different base classifiers, and the base classifier which obtains the best accuracy is selected.

In Fig. 4 an example where a different base classifier is applied for each sub-problem can be seen.

3.1.3. Combination

To take the final decision, we have decided to apply the majority vote; the most voted class is that which is selected. We consider this due to three reasons:

1. As different base classifiers are used for each sub-problem, combination strategies based on the confidence level are not appropriate because each base classifier confidence level is calculated differently.
2. As stated in Galar et al. (2011) the majority vote obtains robust results compared with other combination strategies in OVO.
3. It is the simplest one.

Table 2
Compatibility between OVO and OVA.

	Accuracy of OVO	Accuracy of OVA	Accuracy of OVA when OVO fails	Accuracy of OVO when OVA fails
Abalone	24.946	18.793	12.268	18.910
Annealing	92.383	92.116	25.214	26.508
Arrhythmia	65.885	63.496	29.720	34.319
Balance	78.910	78.910	20.898	20.742
Car	95.914	95.972	34.780	32.725
Cmc	51.460	52.016	24.408	23.597
Dermatology	95.574	91.530	41.702	66.882
Ecoli	81.667	78.571	25.653	35.844
Flare	59.962	56.023	20.195	26.7482
Glass	63.084	60.374	30.566	35.531
Iris	93.467	92.800	1.250	12.500
Nursery	98.622	98.647	6.287	4.172
Optdigits	92.281	87.434	52.325	70.732
Page-blocks	96.824	96.642	17.794	22.202
Pendigits	95.950	94.039	58.541	71.864
Satimage	86.017	83.708	40.617	49.052
Segment	95.039	94.251	45.869	53.228
Vehicle	68.842	68.251	39.939	40.984
Waveform	76.212	74.144	49.187	53.193
Wine	88.202	89.888	54.609	47.018
WineRed	57.486	58.649	29.100	26.977
WineWhite	54.924	53.973	26.774	28.273
Yeast	56.267	55.418	17.459	19.071
Zoo	90.297	90.693	43.222	38.770

$$\begin{array}{cccccccccc}
NB & 3NN & C4.5 & 3NN & C4.5 & SVM & NB & SVM & SVM & 3NN \\
\left(\begin{array}{cccccccccc}
+1 & -1 & -1 & -1 & +1 & +1 & +1 & 0 & 0 & 0 \\
-1 & +1 & -1 & -1 & -1 & 0 & 0 & +1 & +1 & 0 \\
-1 & -1 & +1 & -1 & 0 & -1 & 0 & -1 & 0 & +1 \\
-1 & -1 & -1 & +1 & 0 & 0 & -1 & 0 & -1 & -1
\end{array}\right)
\end{array}$$

Fig. 4. Different base classifier for each sub-problem.

3.2. Experimental setup

In this section we show the experimental results obtained with different databases. We compare our method with other state-of-the-art strategies and we run our experiments over the datasets shown in Table 1.

3.2.1. Classifiers

To carry out our experiments we use some classifiers from WEKA (Waikato Environment for Knowledge Analysis) (Hall et al., 2009).

Among the classifiers that Weka offers, we have selected the following ones:

- Naive Bayes (John & Langley, 1995), statistical learning algorithm. It is based on Bayesian rules and, given that the value of the class is known, it assumes independence between the occurrences of feature values to predict the class.
- J48 (C4.5 clone) (Quinlan, 1993), decision tree algorithm. It makes a post-pruning phase, based on error based pruning algorithm.
- IBK (K-NN clone) (Aha, Kibler, & Albert, 1991), distance based algorithm. An object is classified by a majority vote of its K nearest neighbors. The value of K is set to 3.
- SM0 (SVM clone) (Platt, 1999), kernel methods. It creates a hyperplane where the categories are divided by a clear gap that is as wide as possible.
- JRip (Ripper clone) (Cohen, 1995), rule induction classifier. It builds a rule-set by repeatedly adding rules to an empty rule-set until all positive examples are covered.
- MultilayerPerceptron (Rumelhart, Hinton, & Williams, 1985), an artificial neural network. It is a feedforward network of neurons which map input vectors to output vectors.

As it can be seen, the selected classifiers are from different natures in order to give variability and reliability to the experimental phase. It is worth saying that in our experiments we have used the default parameters of the classifiers.

3.2.2. Strategies summarized

In this sub-section we briefly describe the class-binarization strategies that are used for the comparison. It is worth remembering that in all the strategies the best base classifier is selected for each sub-problem.

- One-vs-All (OVA): Each sub-problem compares one class with the rest of classes. The class with the highest confidence level is selected.
- All-And-One (A&O) (García-Pedrajas & Ortiz-Boyer, 2006; Ko & Byun, 2003): 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.
- Max-Wins (Friedman, 1996; Fürnkranz, 2002): For decomposition OVO is applied: each sub-problem compares two classes between them, ignoring the rest. And the majority vote is used to take the final decision.
- Weighted Voting (WV): The weight for the vote is given by the confidence level of the classifier. The class with the largest sum value is predicted.
- Pairwise Coupling (PC) (Hastie & Tibshirani, 1998): PC tries to find the posterior probability of each class ($p_1 \ldots p_K$) given the posterior probability of all the pairwise sub-problems (r_{ij}). To do so, the problem is transformed into an iterative problem where the Kullback–Leibler distance between r_{ij} and μ_{ij} is minimized ($\mu_{ij} = p_i/(p_i + p_j)$).
- Decision Directed Acyclic Graph (DDAG) (Platt et al., 2000): 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.
- OVA + OVO: Combination of OVA and OVO outputs. The majority vote is used to take the final decision.

3.2.3. Experimental results

In order to give a real perspective, we have applied 5×2 fold cross validation to each database (Demšar, 2006). But firstly each binarization strategy needs a validation process to select the most accurate base classifiers for each binary sub-problem. Therefore, we have applied 5-hold out for each fold, where we have used the 70% as training set and the 30% as testing set.

In Table 3 we show the results obtained for our new method and those obtained with state-of-the-art methods. The best result is highlighted in bold. It can be observed that OVA + OVO obtains the best result in 14 databases. Although OVA + OVO obtains promising results, we have continued with the experiments and in the next Section we will show the main proposal of this paper.

4. Proposed approach: NOV@

In this section we present our new proposal *New One Versus*$_{One}^{All}$. But before explaining the method, we will show the reasons that have led us to develop this proposal.

4.1. Motivation through NOV@

In the previous experiment the results obtained by A&O and OVA methods were lower than expected. Rifkin and Klautau (2004) showed the strength of OVA when the classifier was well-tuned and we consider that selecting the best base classifier for each sub-problem is a good way to tune the sub-problems. Consequently we have made an analysis in order to find the reason for these low results.

4.1.1. The strength of OVA

In order to analyse the behavior of OVA, we have carried out a new experiment. Firstly, we have seen how OVA works by only taking into account the cases where there is only one sub-problem that returns a positive-class prediction. In other words, there is only one case among all the sub-problems in which one class is selected instead of the rest of the classes.

In Table 4 it can be seen the percentage of the cases where OVA takes the final decision under the aforementioned circumstances and which is the accuracy obtained. The results show that a large amount of cases are classified under these circumstances, obtaining high accuracy. Therefore, we deduce that OVA fails with the remaining cases, so they are analysed.

4.1.2. The weakness of OVA

After the analysis, we have deduced that the reason for the bad results of OVA and A&O is because the type of classifier used in

Table 3
Accuracy using the six compared methods and different base classifiers for each sub-problem.

Database	OVA	A&O	Max-Wins	WV	PC	DDAG	OVA + OVO
Abalone	16.553	22.993	**26.598**	26.589	26.426	25.171	**26.598**
Annealing	98.062	97.929	98.196	98.196	98.129	98.085	**98.307**
Arrhythmia	63.850	65.265	69.292	69.292	**71.150**	68.407	71.018
Balance	90.128	89.840	90.096	90.929	90.865	90.641	**91.058**
Car	96.273	95.914	95.787	95.984	95.741	95.845	**96.296**
Cmc	48.350	49.220	52.831	52.492	52.465	51.840	**53.646**
Dermatology	95.519	97.158	97.541	**97.596**	**97.596**	97.541	97.432
Ecoli	82.202	83.452	83.095	**83.988**	83.869	83.631	83.810
Flare	54.540	58.668	**60.094**	59.756	59.587	59.606	**60.094**
Glass	64.206	63.458	64.112	64.486	**65.327**	63.458	65.140
Iris	**95.333**	94.800	94.800	94.800	94.800	94.800	94.933
Nursery	99.336	99.384	99.451	99.477	99.452	99.431	**99.653**
Optdigits	98.456	**98.480**	98.466	98.463	98.470	98.395	98.473
Page-blocks	96.707	96.799	96.751	96.828	96.667	96.777	**96.897**
Pendigits	**99.250**	99.221	99.210	99.214	99.207	99.143	99.238
Satimage	89.958	90.126	90.051	90.058	90.098	89.961	**90.256**
Segment	95.844	96.017	96.104	96.277	96.329	95.983	**96.476**
Vehicle	**80.567**	79.551	78.842	79.362	79.433	79.102	79.905
Waveform	84.152	86.436	**86.636**	**86.636**	**86.636**	**86.636**	86.276
Wine	95.955	96.404	96.404	96.292	96.404	96.404	**96.742**
WineRed	57.949	57.423	57.711	57.761	57.811	57.386	**58.111**
WineWhite	52.748	52.523	54.332	54.075	54.067	53.050	**55.039**
Yeast	56.631	56.536	57.615	57.655	57.251	57.049	**57.803**
Zoo	92.875	92.282	**93.267**	92.875	92.678	93.071	92.875

each sub-problem has a big influence on the final decision. OVA and A&O obtain each class confidence level taking into account only one sub-problem, and in each sub-problem different base classifiers are applied. Each type of classifier uses a different methodology to calculate the confidence level, hence, these confidence levels have different meanings. Some classifiers tend to distribute the confidence level among the classes more equally than others. As a consequence, in the cases where all the output of OVA is negative, the classes obtained using these classifiers are more likely to be selected. We will try to clarify this problem with the following example:

Example: Let us consider a 3-class problem $\{\theta_1, \theta_2, \theta_3\}$ and a new case to be classified. The confidence levels obtained for each OVA sub-problem for classifiers C_1 and C_2 can be seen in Table 5.

Table 4
Accuracy and percentage of the cases where OVA takes the final decision when in only one sub-problem the class is selected instead of the rest of the classes.

	Accuracy	Percentage
Abalone	40.146	0.656
Annealing	99.070	97.996
Arrhythmia	79.207	71.593
Balance	95.746	89.327
Car	98.024	94.005
Cmc	63.399	51.677
Dermatology	98.401	95.574
Ecoli	87.537	88.691
Flare	82.237	47.636
Glass	74.842	69.626
Iris	95.830	99.067
Nursery	99.818	98.307
Optdigits	98.545	99.797
Page-blocks	97.809	97.552
Pendigits	99.393	99.556
Satimage	90.873	97.725
Segment	98.240	94.762
Vehicle	87.301	76.809
Waveform	89.241	83.200
Wine	96.901	96.854
WineRed	63.311	69.206
WineWhite	58.871	65.039
Yeast	67.274	59.973
Zoo	97.613	91.287

It can be observed in Table 5 that in both classifiers, θ_1 obtains the highest confidence levels; as a consequence, θ_1 should be assigned to the new case. Moreover, it is possible to observe that all the classes obtain higher confidence levels with C_1 than with C_2. So, let us consider that after the validation phase, C_2 is selected for $\theta_1 - vs - all$ and C_1 for $\theta_2 - vs - all$ and for $\theta_3 - vs - all$. In this case, θ_2 is the class with the highest confidence level (0.20), so θ_2 is assigned to the new case.

In this example, it can be seen that the classes classified with C_1 have higher probability to be selected than the classes classified with C_2. Hence, the final decision could be different, depending on the type of classifier selected in each sub-problem. That is why we consider it is not fair to compare the confidence levels among them.

In order to avoid this problem, the accuracy obtained for each classifier of the sub-problems in the validation phase, could be used as a confidence level. But we do not consider this appropriate because the accuracy depends on how the classes are distributed. The best differentiated classes are more likely to be selected, because the sub-problems where they take part obtain a higher accuracy in the validation phase.

4.2. New One Versus$^{All}_{One}$ (NOV@)

We have shown that it is not a sound alternative to depend on the confidence levels when different base classifiers are used, moreover we show the strength of OVA in the case that among the sub-problems in only one of them one class outperforms the rest. Furthermore, previously was shown that OVO is able to correct some errors made by OVA.

Table 5
Confidence levels obtained in OVA for each sub-problem for C_1 and C_2 classifiers.

Sub-problem	C_1		C_2	
	θ_i	All	θ_i	All
θ_1-*vs-All*	0.33	0.67	0.17	0.83
θ_2-*vs-All*	0.20	0.80	0.11	0.89
θ_3-*vs-All*	0.07	0.93	0.05	0.95

Table 6
Accuracy using the seven compared methods and different base classifiers for each sub-problem.

Database	OVA	A&O	Max-Wins	WV	PC	DDAG	OVA + OVO	NOV@
Abalone	16.553	22.993	**26.598**	26.589	26.426	25.171	**26.598**	26.560
Annealing	98.062	97.929	98.196	98.196	98.129	98.085	98.307	**98.396**
Arrhythmia	63.850	65.265	69.292	69.292	71.150	68.407	71.018	**72.124**
Balance	90.128	89.840	90.096	90.929	90.865	90.641	**91.058**	90.994
Car	96.273	95.914	95.787	95.984	95.741	95.845	96.296	**96.563**
Cmc	48.350	49.220	52.831	52.492	52.465	51.840	53.646	**53.863**
Dermatology	95.519	97.158	97.541	**97.596**	**97.596**	97.541	97.432	97.377
Ecoli	82.202	83.452	83.095	83.988	83.869	83.631	83.810	**84.643**
Flare	54.540	58.668	**60.094**	59.756	59.587	59.606	**60.094**	59.794
Glass	64.206	63.458	64.112	64.486	65.327	63.458	65.140	**67.009**
Iris	95.333	94.800	94.800	94.800	94.800	94.800	94.933	**95.467**
Nursery	99.336	99.384	99.451	99.477	99.452	99.431	99.653	**99.671**
Optdigits	98.456	**98.480**	98.466	98.463	98.470	98.395	98.473	98.452
Page-blocks	96.707	96.799	96.751	96.828	96.667	96.777	96.897	**97.003**
Pendigits	**99.250**	99.221	99.210	99.214	99.207	99.143	99.238	99.221
Satimage	89.958	90.126	90.051	90.058	90.098	89.961	**90.256**	90.030
Segment	95.844	96.017	96.104	96.277	96.329	95.983	96.476	**96.667**
Vehicle	80.567	79.551	78.842	79.362	79.433	79.102	79.905	**80.922**
Waveform	84.152	86.436	**86.636**	**86.636**	**86.636**	**86.636**	86.276	85.948
Wine	95.955	96.404	96.404	96.292	96.404	96.404	**96.742**	96.629
WineRed	57.949	57.423	57.711	57.761	57.811	57.386	58.111	**58.487**
WineWhite	52.748	52.523	54.332	54.075	54.067	53.050	55.039	**55.300**
Yeast	56.631	56.536	57.615	57.655	57.251	57.049	57.803	**58.679**
Zoo	92.875	92.282	**93.267**	92.875	92.678	93.071	92.875	**93.267**
Mean	79.394	79.995	80.720	80.795	80.852	80.475	81.087	**81.378**
Rank	6.2	6.0	4.6	4.0	4.5	5.8	2.5	**2.3**

Considering these three facts, we propose a new version of OVA where the confidence level is not taken into account; instead the majority vote is used. As previously mentioned, the problem of the majority vote in OVA is that it is common for there to be ties. In this case, the ties are broken by applying OVO only taking into account the tie-classes. We have denoted this new method as *New One Versus*$^{All}_{One}$ (NOV@). When a new case to be classified arrives there are 3 possibilities:

- If only one of the sub-problems gives a positive result, we consider that it is sufficiently reliable, therefore NOV@ returns this class.
- If in more than one case a positive result is obtained, then there is a tie. Hence, Max-Wins is applied only taking into account the tie classes.
- If all the sub-problems obtain a negative result, we consider that OVA has not enough reliability to take the final decision, so Max-Wins is applied with all the classes.

With this new algorithm our aim is to improve OVA's performance. Moreover since the majority of instances are classified applying OVA (Table 4), the new algorithm reduces the number of sub-problems of OVO.

5. Experimental results

We run this new method with the same characteristics as run in the previous experiments (Section 3.2). Table 6 shows the obtained results. The best result is highlighted in bold. It can be seen that NOV@ obtains the best result in 15 of the databases, whereas OVA + OVO obtains the best result in 5. Moreover NOV@ obtains the best mean followed by OVA + OVO.

We have shown in the previous section (Section 4.1) when different base classifiers are being used, care must be taken when strategies that depend on the confidence level are used. However WV and PC also depend on the confidence levels of the sub-problems and, oddly, WV and PC are the state-of-the-art algorithms that obtain the best mean results. The difference between A&O and OVA with WV and PC is that A&O and OVA obtain the confidence level of each class only taking into account one sub-problem, while WV and PC take into account the confidence levels of the different sub-problems. This leads us to think that this combination tends to compensate the confidence levels.

In order to obtain a meaningful conclusions, we carry out statistical analysis to find whether significant differences among the results obtained exists or not. According to García, Fernández, Luengo, and Herrera (2010), we have used the Iman-Davenport test to detect statistical differences among the different strategies. This test rejects the null hypothesis of equivalence between algorithms since p-value (0.0001) is lower than our α-value (0.1). Thus, we have applied Shaffer post hoc test in order to find out which algorithms are distinctive among them. Table 7 shows the most relevant results of the test, where "+" symbol implies that the first algorithm is statistically better than the confronting one, whereas "=" means that there are not significant differences between the compared algorithms. The method having the best performance

Table 7
Shaffer test.

Hypothesis	p-Value
NOV@ vs OVA	+(1.1904E−006)
OVA + OVO vs OVA	+(3.2931E−006)
NOV@ vs A&O	+(5.2983E−006)
NOV@ vs DDAG	+(1.5605E−005)
OVA + OVO vs A&O	+(1.8148E−005)
OVA + OVO vs DDAG	+(5.0998E−005)
NOV@ vs Max-Wins	+(0.0250)
WV vs OVA	+(0.0376)
NOV@ vs PC	+(0.0376)
OVA + OVO vs Max-Wins	+(0.0424)
OVA + OVO vs PC	+(0.0682)
WV vs A&O	+(0.0820)
WV vs DDAG	=(0.1524)
PC vs OVA	=(0.2552)
NOV@ vs WV	=(0.2552)
Max-Wins vs OVA	=(0.3269)
OVA + OVO vs WV	=(0.4067)
PC vs A&O	=(0.4309)
Max-Wins vs A&O	=(0.5548)
PC vs DDAG	=(0.5709)
Max-Wins vs DDAG	=(0.6998)

Table 8
Comparison of the training time of 8 methods (in milliseconds).

Database	OVA	A&O	OVO	WV	PC	DDAG	OVA + OVO	NOV@
Abalone	752,004	2,00,0680	1,248,676	1,248,676	1,248,676	1,248,676	2,000,680	2,000,680
Annealing	168,841	304,570	135,729	135,729	135,729	135,729	304,570	304,570
Arrhythmia	5,401,826	10,538,305	5,136,479	5,136,479	5,136,479	5,136,479	10,538,305	10,538,305
Balance	3782	6611	2829	2829	2829	2829	6611	6611
Car	12,948	22,779	9831	9831	9831	9831	22,779	22,779
Cmc	15,569	25,635	10,066	10,066	10,066	10,066	25,635	25,635
Dermatology	46,394	86,553	40,159	40,159	40,159	40,159	86,553	86,553
Ecoli	6182	13,078	6896	6896	6896	6896	13,078	13,078
Flare	34,020	63,916	29,896	29,896	29,896	29,896	63,916	63,916
Glass	3889	7656	3767	3767	3767	3767	7656	7656
Iris	777	1362	585	585	585	585	1362	1362
Nursery	516,714	857,053	340,339	340,339	340,339	340,339	857,053	857,053
Optdigits	3,734,410	7,035,312	3,300,902	3,300,902	3,300,902	3,300,902	7,035,312	7,035,312
Page-blocks	286,433	534,925	248,492	248,492	248,492	248,492	534,925	534,925
Pendigits	1,731,714	3,190,942	1,459,228	1,459,228	1,459,228	1,459,228	3,190,942	3,190,942
Satimage	1,622,867	2,789,917	1,167,050	1,167,050	1,167,050	1,167,050	2,789,917	2,789,917
Segment	203,956	371,821	167,865	167,865	167,865	167,865	371,821	371,821
Vehicle	33,051	57,489	24,438	24,438	24,438	24,438	57,489	57,489
Waveform	298,047	470,827	172,780	172,780	172,780	172,780	470,827	470,827
Wine	2563	4298	1735	1735	1735	1735	4298	4298
WineRed	48,824	91,505	42,681	42,681	42,681	42,681	91,505	91,505
WineWhite	323,583	591,057	267,474	267,474	267,474	267,474	591,057	591,057
Yeast	51,559	103,720	52,161	52,161	52,161	52,161	103,720	103,720
Zoo	3501	7313	3812	3812	3812	3812	7313	7313
Mean	637,643.92	1,215,721.83	578,077.92	578,077.92	578,077.92	578,077.92	1,215,721.83	1,215,721.83

Table 9
Comparison of the testing time of 8 methods (in milliseconds).

Database	OVA	A&O	OVO	WV	PC	DDAG	OVA + OVO	NOV@
Abalone	2168	108,513	25,320	29,089	167,479	2950	25,897	132,295
Annealing	570	670	1065	1068	1225	437	1633	579
Arrhythmia	5886	6354	19,193	19,255	20,736	3806	25,061	9873
Balance	24	44	34	35	127	42	50	24
Car	48	94	352	357	547	220	399	68
Cmc	28	58	51	51	140	31	76	43
Dermatology	256	394	338	342	440	110	588	257
Ecoli	182	234	289	294	484	83	470	203
Flare	139	275	386	392	676	125	498	315
Glass	35	44	45	47	103	25	77	46
Iris	10	17	12	12	55	14	21	12
Nursery	804	1023	1379	1409	3686	616	2162	817
Optdigits	579,849	1,204,315	646,695	646,861	652,546	107,932	1,226,422	581,075
Page-blocks	1805	2030	3705	3718	4675	2971	5503	1856
Pendigits	403,145	673,507	556,751	557,102	568,207	84,669	959,782	404,812
Satimage	149,632	241,427	138,126	138,159	139,908	51,400	287,481	151,562
Segment	2248	2381	3312	3331	4233	1046	5546	2322
Vehicle	71	98	129	129	219	64	198	85
Waveform	372	567	309	317	823	225	661	397
Wine	39	57	40	41	54	22	76	35
WineRed	149	230	513	531	1,186	236	656	314
WineWhite	5148	5322	5521	5563	7524	1658	10,650	6971
Yeast	1028	1082	1859	1899	3386	478	2871	1892
Zoo	46	50	105	108	149	23	148	48
Mean	48,070.08	93,699.42	58,563.71	58,754.58	65,775.33	10,799.29	106,538.58	53,995.88

is NOV@, closely followed by OVA + OVO. Both methods significantly improve all the remaining strategies, except WV. However, it can be seen that our proposed methods obtain more robust results since WV only outperforms significantly OVA and A&O. Moreover if we compare the rank of NOV@, OVA + OVO and WV, our two methods obtain more stable results.

5.1. Computational load

In order to measure the computational cost and complexity of our proposals, in Tables 8–10 we show the training and testing times and the number of binary classifiers used in each strategy.

Table 8 shows the training time of each strategy for the different databases. It is observed that the strategies are divided into three groups: OVA, OVO aggregations (Max-Wins, WV, PC and DDAG) and combinations of OVA and OVO (A&O, OVA + OVO and NOV@). It can be seen that the combinations of OVA and OVO need more training time. On the other hand, the classification time needed by OVA is slightly longer than OVO aggregations for problems with few classes. Although OVA uses less sub-problems than OVO, the size of the sub-problems is higher in OVA and that is why the time required to train the classifier is longer. However, it can be noted that when the number of classes is high, the training time of OVA is shorter than in OVO.

Table 10
Number of classifiers used by different methods.

Database	OVA	AØ	OVO	WV	PC	DDAG	OVA + OVO	NOV@
Abalone	28	320.153	378	378	378	27	406	389.936
Annealing	5	6	10	10	10	4	15	5.119
Arrhythmia	13	14.221	78	78	78	12	91	27.596
Balance	3	4.314	3	3	3	2	6	3.298
Car	4	5.063	6	6	6	3	10	4.250
Cmc	3	4.015	3	3	3	2	6	4.386
Dermatology	6	11.986	15	15	15	5	21	6.404
Ecoli	8	11.390	28	28	28	7	36	9.746
Flare	6	11.003	15	15	15	5	21	13.798
Glass	6	7.133	15	15	15	5	21	9.222
Iris	3	4.317	3	3	3	2	6	3.015
Nursery	5	6.013	10	10	10	4	15	5.109
Optdigits	10	53.345	45	45	45	9	55	10.090
Page-blocks	5	6.001	10	10	10	4	15	5.147
Pendigits	10	32.019	45	45	45	9	55	10.147
Satimage	6	15.733	15	15	15	5	21	6.248
Segment	7	8.085	21	21	21	6	28	7.551
Vehicle	4	5.001	6	6	6	3	10	4.792
Waveform	3	4.518	3	3	3	2	6	3.303
Wine	3	4.825	3	3	3	2	6	3.056
WineRed	6	7.802	15	15	15	5	21	9.574
WineWhite	7	8	21	21	21	6	28	12.795
Yeast	10	11	45	45	45	9	55	27.313
Zoo	7	9.277	21	21	21	6	28	8.570
Mean	7.052	23.801	33.288	33.288	33.288	5.833	40.288	24.603

Table 9 shows the testing time of each strategy for the different databases. It can be seen that NOV@ tends to be one of the fastest among the 8 methods. Only DDAG and OVA tend to be faster closely followed by A&O. Moreover except for in Abalone database, NOV@ spends a bit more time than OVA. Immediately below them on the table are OVO and WV that need similar time, while PC needs slightly more time since it follows an iterative procedure. Finally, OVA + OVO tends to be the slowest method.

Table 10 shows the mean number of classifiers needed in each strategy for the different databases. The obtained results tend to be similar to those found in Table 9. However, this time NOV@ needs slightly more classifiers than A&O.

Viewing the results obtained in Tables 8–10, although NOV@ is one of the slowest strategies to train, it is one of the fastest strategies at classification time, only outperformed by OVA and DDAG. On the other hand, OVA + OVO is the slowest strategy.

6. Conclusion

This paper has presented a new approach to combine pairwise classifiers which aims to improve classification accuracy in supervised classification multi-class problems. Starting from a single combination of two well-known approaches (One-vs-All and One-vs-One), a new procedure to make a classifier combination is presented, NOV@, in which both OVA and OVO are combined in a different way than found in the rest of literature. The results obtained by the new approach on different datasets are subjected to in-depth analyses and compared with those of the most used state-of-the-art methods. From the comparison, it is shown that the results are very competitive, ranking in the first position from the accuracy point of view, and among the best in classification time; this last due to a low number of classifiers.

It has also been shown that OVA and OVO strategies are compatible and can be combined with each other, even when different base classifiers are used for each sub-problem. This is possible because each sub-problem has been tackled as an independent one, and hence it is treated as a new classification problem in which two classes are to be discriminated. The proposed methods – the single one, OVA + OVO, and NOV@ – have been implemented and tested over 20 databases from the UCI repository, obtaining significant improvements over other state-of-the-art strategies. In addition to this, the two methods maintain the simplicity that has made of OVA and OVO the most used class-binarization methods. Furthermore, comparing our methods with other state-of-the-art algorithms, we have made an empirical study in order to analyse how different class-binarization strategies work when different base classifiers are applied in each sub-problem.

A further analysis of the computational load of the used approaches has shown that the proposed approach – NOV@ – has competitive classification times compared with the state-of-the-art approaches, and that it has a lower computational cost with respect to the most powerful classifiers. When training time is compared, though, the results were worse than those of other approaches. However, this was expected given that all the sub-problems decomposed by OVA and OVO have to be trained.

As future works, we are planning to apply more single classifiers in the best classifier selection phase as well as other approaches to combine the different results. In this sense, the method proposed by Polat (2013) seems to set the right direction for future experiments.

On the other hand, real applications of the proposed model are to be analysed. We are going to test the approach with real problems related data, for example we are trying to obtain new data related to our previous work (Ferreiro, Arnaiz, Sierra, & Irigoien, 2012) in order to carry on with the experiments. Other application of the proposed approach can be website phishing, which is considered one of the crucial security challenges for the online community due to the massive numbers of online transactions performed on a daily basis. We are also planning to perform an experiment similar to the one proposed by Abdelhamid, Ayesh, and Thabtah (2014).

Acknowledgments

The work described in this paper was partially conducted within the Basque Government Research Team Grant and the

University of the Basque Country UPV/EHU and under Grant UFI11/45 (BAILab). I. Mendialdua holds a Grant from Basque Government.

References

Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based associative classification data mining. *Expert Systems with Applications*.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning, 6*(1), 37–66.

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

Bache, K., & Lichman, M. (2013). *UCI machine learning repository*. Irvine, CA: University of California, School of Information and Computer Science.

Bagheri, M. A., Gao, Q., & Escalera, S. (2012). Efficient pairwise classification using local cross off strategy. In *Advances in Artificial Intelligence* (pp. 25–36). Springer.

Chen, J., Wang, C., & Wang, R. (2009). Adaptive binary tree for fast SVM multiclass classification. *Neurocomputing, 72*(13), 3370–3375.

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the 12th international conference on machine learning* (pp. 115–123). Morgan Kaufman.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research, 7*, 1–30.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2.

Fei, B., & Liu, J. (2006). Binary tree of SVM: A new fast multiclass training and classification algorithm. *IEEE Transactions on Neural Networks, 17*(3), 696–704.

Ferreiro, S., Arnaiz, A., Sierra, B., & Irigoien, I. (2012). Application of Bayesian networks in prognostics for a new integrated vehicle health management concept. *Expert Systems with Applications, 39*(7), 6402–6418.

Friedman, J. (1996). Another approach to polychotomous classification. Tech. Rep., Technical report, Stanford University, Department of Statistics.

Fürnkranz, J. (2002). Round robin classification. *The Journal of Machine Learning Research, 2*, 721–747.

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*(8), 1761–1776.

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*(12), 3412–3424.

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*(10), 2044–2064.

García-Pedrajas, N., & Ortiz-Boyer, D. (2006). Improving multiclass pattern recognition by the combination of two strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(6), 1001–1006.

García-Pedrajas, N., & Ortiz-Boyer, D. (2011). An empirical study of binary classifier fusion methods for multiclass classification. *Information Fusion, 12*(2), 111–130.

Ghaffari, H. R., & Yazdi, H. S. (2013). Multiclass classifier based on boundary complexity. *Neural Computing and Applications*, 1–9.

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*(1), 10–18.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*(10), 993–1001.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. *The Annals of Statistics, 26*(2), 451–471.

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks, 13*(2), 415–425.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th conference on uncertainty in artificial intelligence* (pp. 338–345). Morgan Kaufman Publishers Inc..

Ko, J., & Byun, H. (2003). Binary classifier fusion based on the basic decomposition methods. In *Proceedings of the fourth international conference on multiple classifier systems* (pp. 146–155). Springer.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning, 51*(2), 181–207.

Lebrun, G., Lezoray, O., Charrier, C., & Cardot, H. (2007). An ea multi-model selection for SVM multiclass schemes. In *Proceedings of the ninth international work conference on artificial neural networks (IWANN07)* (pp. 260–267). Springer.

Liepert, M. (2003). Topological fields chunking for German with SVMs: Optimizing SVM-parameters with GAs. In *Proceedings of the international conference on recent advances in natural language processing*.

Liu, B., Hao, Z., & Yang, X. (2007). Nesting algorithm for multi-classification problems. *Soft Computing, 11*(4), 383–389.

Lorena, A. C., & de Carvalho, A. C. (2010). Building binary-tree-based multiclass classifiers using separability measures. *Neurocomputing, 73*(1618), 2837–2845.

Lorena, A. C., de Carvalho, A. C., & Gama, J. M. (2008). A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review, 30*(1–4), 19–37.

Madzarov, G., & Gjorgjevikj, D. (2009). Multi-class classification using support vector machines in decision tree architecture. In *IEEE EUROCON 2009, EUROCON'09* (pp. 288–295). IEEE.

Moreira, M., & Mayoraz, E. (1998). Improved pairwise coupling classification with correcting classifiers. In *Proceedings of the 10th european conference on machine learning (ECML-98)* (pp. 160–171). Springer.

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

Platt, J. C., Cristianini, N., & Shawe-Taylor, J. (2000). Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems, 12*(3), 547–553.

Polat, K. (2013). Data weighting method on the basis of binary encoded output to solve multi-class pattern classification problems. *Expert Systems with Applications, 40*(11), 4637–4647.

Polat, K., & Güneş, S. (2009). A novel hybrid intelligent method based on C4.5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Systems with Applications, 36*(2, Part 1), 1587–1592.

Pujol, O., Radeva, P., & Vitria, J. (2006). Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 0162-8828, *28*(6), 1007–1012.

Quinlan, J. R. (1993). *C4. 5: Programs for machine learning* (Vol. 1). Morgan Kaufmann.

Reid, S. R. (2010). Model combination in multiclass classification (Ph.D. thesis). University of Colorado.

Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. *The Journal of Machine Learning Research, 5*, 101–141.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. Tech. Rep., DTIC Document.

Szepannek, G., Bischl, B., & Weihs, C. (2009). On the combination of locally optimal pairwise classifiers. *Engineering Applications of Artificial Intelligence, 22*(1), 79–85.

Tang, E., Suganthan, P., & Yao, X. (2006). An analysis of diversity measures. *Machine Learning, 65*(1), 247–271.

14

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

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

Authors: I. Mendialdua, G. Echegaray, I. Rodriguez, E. Lazkano and B. Sierra.

Journal: Under major revision in Neurocomputing.
Submitted in June 2014
Revision submitted in February 2015

Publisher: Elsevier

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.

$$classes \left\{ \begin{array}{c|cccccc} & \multicolumn{6}{c}{\overbrace{\hphantom{+1\ \ 0\ \ -1\ \ -1\ \ 0\ \ +1}}^{classifiers}} \\ & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\ \theta_1 & +1 & 0 & -1 & -1 & 0 & +1 \\ \theta_2 & +1 & +1 & -1 & -1 & +1 & 0 \\ \theta_3 & -1 & +1 & +1 & -1 & 0 & 0 \\ \theta_4 & 0 & -1 & 0 & +1 & 0 & +1 \\ \theta_5 & -1 & -1 & 0 & -1 & -1 & -1 \end{array} \right. \qquad \begin{array}{l} f_1 \rightarrow \theta_1, \theta_2 \ vs \ \theta_3, \theta_5 \\ f_2 \rightarrow \theta_2, \theta_3 \ vs \ \theta_4, \theta_5 \\ f_3 \rightarrow \theta_3 \ vs \ \theta_1, \theta_2 \\ f_4 \rightarrow \theta_4 \ vs \ \theta_1, \theta_2, \theta_3, \theta_5 \\ f_5 \rightarrow \theta_2 \ vs \ \theta_5 \\ f_6 \rightarrow \theta_1, \theta_4 \ vs \ \theta_5 \end{array}$$

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.

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

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.

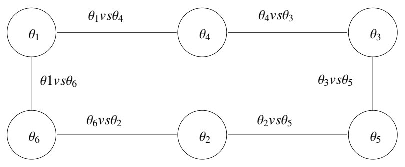
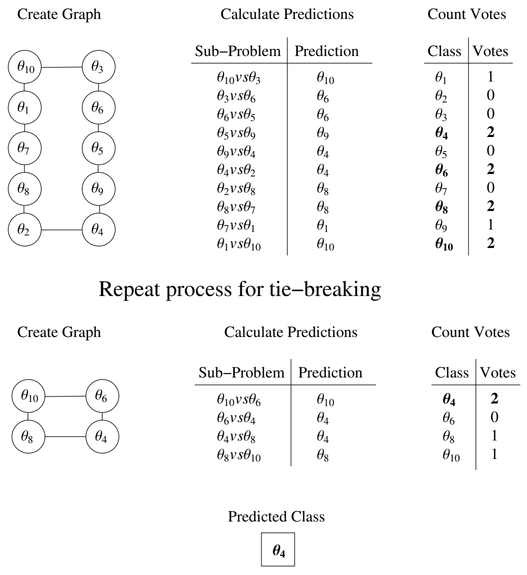


Figure 3: Example of the structure of DUCG for a 6-class problem

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

$$S' = (\theta_4,) \qquad \left(\begin{array}{ccccc} - & 0.25 & 0.25 & 0.09 & 0.41 \\ 0.25 & - & 0.25 & 0.33 & 0.17 \\ 0.25 & 0.25 & - & 0.33 & 0.17 \\ \hline \multicolumn{1}{|c}{0.09} & 0.33 & 0.33 & - & \multicolumn{1}{c|}{0.25} \\ \hline 0.41 & 0.17 & 0.17 & 0.25 & - \end{array}\right)$$

(a) Step 1

$$S' = (\theta_4, \theta_3) \qquad \left(\begin{array}{ccccc} - & 0.27 & 0.27 & - & 0.45 \\ 0.37 & - & 0.37 & - & 0.25 \\ \hline \multicolumn{1}{|c}{0.37} & 0.37 & - & - & \multicolumn{1}{c|}{0.25} \\ \hline - & - & - & - & - \\ 0.55 & 0.22 & 0.22 & - & - \end{array}\right)$$

(b) Step 2

$$S' = (\theta_4, \theta_3, \theta_5) \qquad \left(\begin{array}{ccccc} - & 0.38 & - & - & 0.62 \\ 0.60 & - & - & - & 0.40 \\ - & - & - & - & - \\ - & - & - & - & - \\ \hline \multicolumn{1}{|c}{0.71} & 0.29 & - & - & \multicolumn{1}{c|}{-} \\ \hline \end{array}\right)$$

(c) Step 3

$$S' = (\theta_4, \theta_3, \theta_5, \theta_1, \theta_2) \qquad \begin{pmatrix} - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \\ - & - & - & - & - \end{pmatrix}$$

(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.

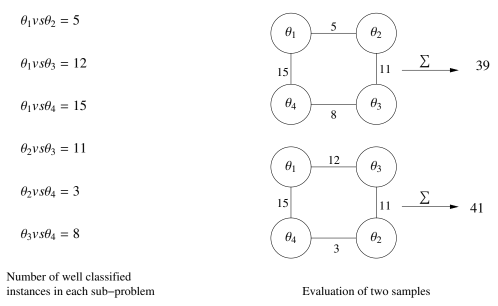

$\theta_1 vs \theta_2 = 5$
$\theta_1 vs \theta_3 = 12$
$\theta_1 vs \theta_4 = 15$
$\theta_2 vs \theta_3 = 11$
$\theta_2 vs \theta_4 = 3$
$\theta_3 vs \theta_4 = 8$
Number of well classified instances in each sub–problem
θ_1
θ_2
θ_3
θ_4
5
11
8
15
Σ
39
12
3
41
Evaluation of two samples

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 signicant 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 signicant 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 NaturePPSN 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.

15

Dynamic selection of the best base classier in One versus One

Title: Dynamic selection of the best base classier in One versus One

Authors: I. Mendialdua, J. M. Martínez-Otzeta, I. Rodriguez, T. Ruiz-Vazquez and B. Sierra.

Journal: Under major revision in Knowledge-Based Systems
Submitted in October 2014
Revision submitted in February 2015

Publisher: Elsevier

Dynamic selection of the best base classifier in One versus One

I. Mendialdua[a,*], J. M. Martínez-Otzeta[a], I. Rodriguez[a], T. Ruiz-Vazquez[b], B. Sierra[a]

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

Abstract

Class Binarization strategies decompose the original multi-class problem into several binary sub-problems. One versus One (OVO) is one of the most popular Class Binarization techniques, which considers every pair of classes as a different sub-problem. Usually, the same classifier is applied to every sub-problem and then all the outputs are combined by some voting scheme. In this paper we present a novel idea where for each test instance we try to assign the best classifier in each sub-problem of OVO. To do so, we have used two simple Dynamic Classifier Selection (DCS) strategies that have not been used in this context. The two DCS strategies use K-NN to obtain the local region of the test-instance, and the classifier that performs the best for those instances in the local region, is selected to classify the new test instance. The difference between the two DCS strategies remains in the weight of the instance. In this paper we also have proposed a novel approach in those DCS strategies. Instead of using the K-NN method to achieve the local regions, we propose to use a version of K-NN obtained from the state-of-the-art called K-NN Equality (K-NNE). K-NNE is similar to K-NN, but it obtains the K nearest neighbors of each class. We have carried out an empirical study over several UCI databases, which shows the robustness of our proposal.

Keywords: Machine Learning, Supervised Classification, Decomposition Strategies, One against One, Classifier Combination, Dynamic Classifier Selection

1. Introduction

The objective of the Supervised Classification strategies is to classify the new unlabelled samples in their correct class. To do so, these strategies create a prediction model (also denoted as classifier) based on a training set of well labelled instances.

A classification problem with only two classes is known as a binary classification problem. A simple example of a binary classification problem are the yes/no or true/false problems. On the other hand the problems with more than two classes are known as multi class problems. However for several kind of classifiers, such as SVM, it is easier to build a classifier to distinguish only between two classes. Because of that, two general approaches have been adopted to deal with multi class problems: to create a single decision function that considers all the classes or to decompose the problem into several binary sub-problems (also known as class-binarization).

In the latest years the class-binarization strategies are getting more common in the literature. There are 3 main techniques: One versus All (OVA)[2], One versus One (OVO)[12] and Error Correcting Output Codes (ECOC)[9]. In this work we focus our attention on OVO strategy, which compares the cases belonging to two classes in each sub-problem; the remaining classes are ignored in each sub-problem.

[*]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)

OVO gives the option to consider each sub-problem as independent and to select a different base classifier in each sub-problem, which could be considered as an example of static classifier selection problem. For classification selection scheme two categories exist: static and dynamic. In the first case, regions of competence are defined during the training phase, while in the second case, they are defined during the classification phase taking into account the characteristics of the sample to be classified.

In the literature it is possible to find several works that propose the selection of different base classifiers in each sub-problem statically; however conclusions of these works are contradictory: some works obtain significant improvements, while others reject this hypothesis.

In this paper, we propose to extend this idea trying to assign dynamically the best base classifier in each sub-problem of OVO. We have called to this new approach DYNOVO. We present several variations of DYNOVO using two simple Dynamic Classifier Selection (DCS) strategies from the state-of-the-art. Those strategies select the classifier that obtains the best accuracy in a local region, which is defined by the K-Nearest Neighbor (K-NN) algorithm. In order to adapt those DCS strategies we have made several changes on the K-NN algorithm, moreover we propose the use of another K-NN version called K-Nearest Neighbor Equality (K-NNE) from the state-of-the-art which fits properly in this problem. For our experiments we have chosen several well-known classifier from the Machine Learning paradigms: SVM, C4.5, Ripper, Naive Bayes and Bayesian Network. We have carried out our experiments over 22 UCI databases. Experimental results show that DYNOVO obtains very good results.

The rest of the paper is organized as follows: In Section 2 we review the Class Binarization techniques, focusing on OVO strategy. In Section 3 we review the Dynamic Classifier Selection technique while Section 4 is devoted to related work. Section 5 describes the proposed approach and Section 6 shows the experimental results obtained. Finally, Section 7 states the conclusions of our work and future research lines.

2. Class Binarization

Several machine learning techniques, such as SVM, were designed to solve two-class problems. However many real-word problems involve the discrimination of more than two classes. In order to use those algorithms in multi-class problem the class binarization strategies divide the original problem into several two-class problems. It has been proven the benefits to use the binarization techniques in multi-class problems [15] and due to those promising results the use of these strategies has been extended to other base classifiers, such as Ripper [14] or C4.5 [9]. In the recent years the class binarization strategies are receiving more attention in the literature, and one indicative of that is that recently several reviews have been published [29] [18] [15].

The Class Binarization techniques are divided by two steps: decomposition and combination.

In the decomposition step, the multi-class problem is decomposed into several binary sub-problems. The most popular strategies consist on grouping classes into two groups in each sub-problem, in this way each binary classifier compares two groups of classes between them. The code-matrix is an easy way to represent how the classes are grouped.

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

Each of these sub-problems returns an output with a prediction. The combination step consists on combining these predictions to made the final decision. The simplest combination is the majority vote, where each sub-problem returns a vote and the class with the largest number of votes is predicted.

Different decomposition strategies have been developed where One Vs One (OVO) is one of the strategies that has received more attention in the literature.

2.1. One versus One (OVO)

OVO decomposition scheme decomposes a K class multiclass problem into a $K(K-1)/2$ sub-problems. Each sub-problem is responsible to differentiate one pair of classes (θ_i, θ_j), where $\theta_i \neq \theta_j$; the remaining classes are ignored.

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

Figure 1: Example of a code matrix

Figure 2 illustrates a code matrix of how a 5-class problem is decomposed in OVO: in each sub-problem one class is represented as +1 class, another one is represented as -1 and the remaining classes are represented as 0.

$$
\begin{pmatrix} +1 & +1 & +1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & +1 & +1 & +1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}
$$

Figure 2: OVO code-matrix

There are different aggregations of combining the output predictions of the sub-problems. The simplest combination strategy is the majority vote [14] [12]. An immediate extension is the Weighted Voting, where the vote of each output is weighted based on the confidence level returned by the classifier [22]. Hastie and Tibshirani [21] propose another combination that tries to find the best approximation of the class posterior probabilities given the posterior probabilities of the pairwise sub-problems.

Although OVO requires a high number of sub-problems (specially when the number of classes is high), it is worth mentioning that each classifier is trained only with the samples from the corresponding pair of classes, hence the decision boundaries to distinguish the classes are simpler and the required time is not high. However there are several proposals that try to reduce the number of sub-problems, where most of these works are based on a hierarchical structure [32] [11].

3. Dynamic Classifier Selection (DCS)

As different classifiers usually make different error on different samples, Dynamic Classifier Selection (DCS) based methods attempt to predict the single classifier which is most likely to be correct for a given sample. To do so, the best classifier for each partition is determined on a validation process. For classification, an unknown sample is assigned to a partition, and the output of the best classifier for that partition is the one used to make the final decision.

The first Dynamic Classification approaches are introduced by Woods [39] and are based on K-NN algorithm. He proposes two methods: Overall Local Accuracy (OLA) and Local Class Accuracy: both methods obtain the classifiers' accuracy in local regions in the surroundings of the unknown test sample, the classifier with the best accuracy is selected to classify the unknown sample. Smith [36] proposes an immediate extension of OLA applying the Distance Weighted K-NN (DW-OLA). Giacinto and Roli [19] also extend Woods's work incorporating distance weighted and classifiers confidence levels to two new methods called A Priori and A Posteriori. On the other hand, there are also other works which are not based on the K-NN method, for instance, Liu and Yuan [28] propose to use clustering: they divide the feature space into several clusters for each base classifier. The unknown sample is assigned to a cluster for each base classifier, and the classifier of the most accurate cluster is selected to classify the unknown sample.

Recently, the DCS methods have been extended to Dynamic Ensemble Selection (DES): instead of finding the most suitable classifier, the most suitable ensemble for each sample is selected. Ko et al. [25] propose 4 new dynamic selection schemes. Those methods obtain the K nearest neighbors of the test point and the classifiers that classify correctly those neighbors, are used as ensemble to classify the test instance. On the other hand, Dos Santos et al. [10] introduce a two step DES method: in the first step, highly accurate candidates ensembles are selected; in the second step, among those ensembles, for each test sample, the ensemble with the largest confidence level is selected. In a further work Cavalin et al. [7] extend the previous work and they adapt it to Dynamic Multistage Organization strategy.

4. Related Works

In a classification problem, the classical way is to select the optimal base classifier for the database and all the sub-problems are classified with this classifier. As in binarization strategies there are too many sub-problems, it is possible that this base classifier could have difficulties to deal with all the sub-problems appeared, returning the wrong result in some of them. This raises the question - should the same base classifier be used on all sub-problems? or should sub-problems be tuned independently?

In the literature there are several works that treat the sub-problems independently. But to our knowledge excepting the introduced by Arruti et al [3] and Bautista et al [6] there are not more works that present an algorithm specifically for the cases that the sub-problems are treated independently. The majority of the approaches propose a method that try to select the best classifier or best parameters of the classifier for each sub-problem and they compare the new proposal with the results obtained without tuning.

On the one hand, some proposals focus on attempting to select the best base classifier in each sub-problem [38]. On the other hand, other approaches try to select the best hyper-parameters of SVM in each sub-problem. Because of the high number of possible values of the hyper-parameters, most of these works use evolutionary algorithms. Lebrun et al [26] and Liepert [27] propose the use of Genetic Algorithms while Souza et al [37] propose the use of Particle Swarm Optimization. The results obtained by these four works are contradictory since two of them consider that the independent tune of the sub-problems is better while the other two consider that there is no significant difference.

Lorena and Carvalho [30] consider that none of the mentioned works perform a rigorous statistical analysis. Thus, they investigate the use of Genetic Algorithms to automatically tune the parameters of each binary SVM. They conclude that the use of same parameter values in all binary SVM is sufficient to obtain good results.

In his Phd Thesis Reid [34] also deals with this problem and he concludes that it is better to tune the classifiers when the decision boundaries of sub-problems have different shape, otherwise, he concludes that it is better the same base classifier.

In the literature we have found an algorithm, proposed by Galar et al. [16] and Bagheri et al. [5] independently, that combine OVO with DCS strategies. Their main idea is to reduce the number of classifiers in OVO avoiding the no competent pairwise comparisons. The K nearest neighbors of a new instance are obtained and OVO is applied only considering those classes which are in the neighborhood.

On the other hand, there is also another work proposed by Kapp et al. [24] that selects the hyper-parameters of SVM dynamically. But this work does not use the DCS strategies and does not treat each sub-problem independently; it is oriented to data-streaming and similar problems. The authors consider that when knowledge about the environment is updated with new observations, the previously parametrized models need to be re-evaluated. To do so, they use the Particle Swarm Optimization.

5. Proposed approach: Dynamic Classifier Selection in OVO (DYNOVO)

Most of the works mentioned in Section 4 follow a similar procedure: they tune statically the classifier of each pairwise sub-problem. The hypothesis that the previous works follow is that the boundaries that distinguish the different sub-problems vary depending on the classes. We extend this idea and we consider that the shape of the boundaries between two classes can vary also, hence the use of the different base classifiers can be appropriate. Because of that, we propose a new method, called DYNOVO, that tries to select the best base classifier dynamically for each test pattern in each binary sub-problem: basically our method combines OVO with Dynamic Classifier Selection (DCS) strategies.

The structure of DYNOVO is similar to those most common DCS strategies and it is divided into two levels: validation and classification. The only difference is that the method is adapted to the pairwise decomposition strategy format.

The aim of the validation step is to see with which base classifier each training instance obtains correct or incorrect results by the different sub-problems. Each training sample is classified by different base classifiers for the different pairwise sub-problems. But instead of classifying it for every sub-problem, it is classified only in those sub-problems where the class the training sample belongs to is distinguished, since the remaining sub-problems can not return the correct result: if the training set belongs to class θ_i, the sub-problem that distinguish θ_j and θ_k ($\theta_i \neq \theta_j, \theta_i \neq \theta_k$) never will return the correct class. Hence, these sub-problems don't need to be treated and computational load is saved in the validation phase.

In the classification step, when an instance to be classified is arrived, our method tries to select the best base classifier for each sub-problem. To do so each sub-problem is treated independently. In each sub-problem the surrounding training samples of the new instance are obtained and the base classifier that obtains the best results for these instances in the validation phase is selected to classify it. In order to make this selection we have chosen the following DCS strategies:

- Overall Local Accuracy (OLA) [39]: When an instance to be classified is arrived, its surrounding region is selected obtaining its K-Nearest Neighbors. It calculates the local accuracy of each base classifier for these K neighbors. The base classifier with the highest local accuracy is selected to classify the test sample.

- Distance Weighted - Overall Local Accuracy (DW-OLA) [36]: This method is an immediate extension of OLA. When the local accuracy is calculated, each K neighbor receives a weight depending on their distance to the test sample, where the closer ones receive a higher weight.

Both strategies use K-NN method to delimit the local region. As we have mentioned previously, our approach has to be adapted to the OVO strategy. Because of that we have made a little change in the K-NN algorithm when the local region is obtained. Moreover we also propose to use a K-NN variation presented in the state-of-the-art called K-NN Equality (K-NNE) [35].

- K Nearest Neighbor (K-NN) [1]: K-NN is one of the most popular machine learning algorithms. When a new instance to be classified is arrived, the K most similar training instances are obtained and the most represented class among those K neighbors is assigned to the new instance. In order to measure the similarity, it is necessary to use a metric, being the euclidean distance one of the most common.

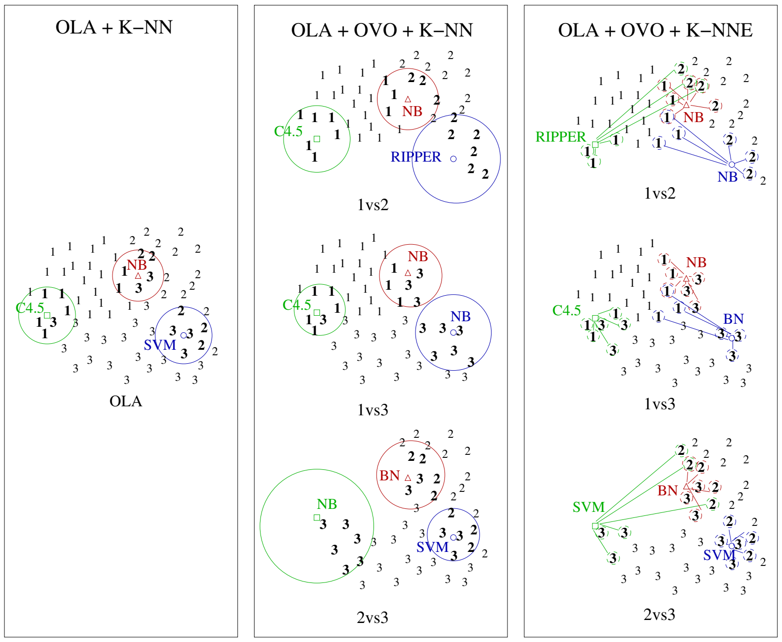
  It is worth mentioning that in this case K-NN is not used to classify a new unlabelled instance, but to delimit the local region of it. Our method tries to select the best base classifier for each sub-problem; because of that each sub-problem is treated independently. Therefore instead of taking into account all the training instances, for each sub-problem it only finds the K nearest neighbors of the test sample that belong to the classes of the sub-problem.

- K Nearest Neighbor Equality (K-NNE) [35]: K-NNE is an extension of K-NN in which the classes are treated independently: it searches in each class the K nearest neighbors and assigns the class whose K neighbors have the minimal mean distance to the sample test. In this way all the classes take part in the final decision.

5.1. Example: obtaining the local regions

Figure 3 illustrates how the local-regions are obtained for 3 new cases in a 3-class problem with different strategies: OLA and our proposal applying K-NN and K-NNE. In the figures the 3 new cases are represented as □, △ and ○. We want to emphasize that the local regions are not used to classify the unlabelled instances as in K-NN, indeed they are used to select the classifier which will be used to classify the unlabelled instances.

In Figure 3(a) it is shown an example of how OLA method obtains the local region applying K-NN method for the 3 unknown samples; in this example the K parameter is given a value of 6. The circle around the new case and with the same color represents its local region, and the 6 nearest neighbors are highlighted in bold. It can be seen that a different base classifier is selected for each of those samples.



(a) Obtaining the local regions of OLA (b) Obtaining the local regions of OLA in OVO using K-NN (c) Obtaining the local regions of OLA in OVO using K-NNE

Figure 3: Example of how the local regions are obtained for each strategy.

Figure 3(b) shows the extension of OLA to OVO. It can be seen that in each sub-problem different samples take part in the decision of the base classifier, hence, in some cases different base classifier are selected to classify the same unlabelled instance in each sub-problem.

Figure 3(c) illustrates an extension of the previous figure using in this case K-NNE to select the local regions, in this approach the value of K is 3. The way to represent the local region of each new case varies: they are formed by the 3 nearest neighbors of each class. The instances that correspond to the local region of each new case are connected by a line of their color and they are highlighted in bold with a circle around them. Comparing with the previous example more distant instances take part in the classifier selection decision, but both classes are in equal conditions.

6. Experiments

In this section we explain the experimental setup. Moreover we carry out an empirical study in order to analyzed the usefulness of DYNOVO. To do so we compare the proposed variations of DYNOVO with the state-of-the-art methods.

6.1. Datasets

We have selected 22 databases from the UCI repository [4] to perform the experiments. A summary of these databases is shown in Table 1.

Table 1: Characteristics of the databases

Database	#Cases	#Atributes	#Classes
Annealing	798	38	5
Balance-scale	625	4	3
Car	1728	6	4
Cmc	1473	9	3
Dermatology	366	33	6
Ecoli	336	7	8
Glass	214	9	7
Image Segmentation	2310	19	7
Iris	150	4	3
Lymph	148	18	4
Nursery	12960	8	5
Optdigits	5620	64	10
Page-blocks	5473	10	5
Pendigits	10992	16	10
Satimage	6435	36	6
Solar-flare-1	323	12	6
Solar-flare-2	1066	12	6
Vehicle	846	18	4
Vowel	990	13	11
Waveform	5000	21	3
Wine	178	13	3
Zoo	101	17	7

6.2. Base Classifiers

To carry out the experiments we have chosen 5 different base classifiers from a software package for Machine Learning Called WEKA [20]. The selected classifiers are from different natures in order to give variability and reliability to the experimental phase. It is worth saying that in our experiments we have treated the classifiers as black boxes and we have used their WEKA package default parameters.

- J48 (C4.5 clone) [33], decision tree algorithm. It makes a post-pruning phase, based on error based pruning algorithm. The parameters used are the following:
  - Confidence Factor = 0.25.
  - Minimum number of instances = 2.
  - Unpruned = False.

- SM0 (SVM clone) [31], kernel methods. It creates a hyperplane where the categories are divided by a clear gap that is as wide as possible. The parameters used are the following:
  - Fit logistic models = False.
  - C = 1.0.
  - Epsilon = 1.0E-12.
  - Kernel = Polynomial kernel.
  - Tolerance parameter = 0.001.
- JRip (Ripper clone) [8], rule induction classifier. It builds a rule-set by repeatedly adding rules to an empty rule-set until all positive examples are covered. The parameters used are the following:
  - Check error rate: True.
  - Minimal weights of instances: 2.0.
  - Number of runs of optimizations: 2.
  - Prune: True.
- Naive Bayes [23], statistical learning algorithm. It is based on Bayesian rules and, given that the value of the class is known, it assumes independence between the occurrences of feature values to predict the class.
- Bayesian Network, [13] statistical learning algorithm. It is a probabilistic graphical model that represents a set of random variables and their conditional independences via a directed acyclic graph. The parameters used are the following:
  - Estimator: Simple Estimator.
  - Search Algorithm: K2.
  - ADTree: False.

6.3. Experimental setup

The classification performance is obtained by means of a stratified 10-fold cross-validation. Some of the compared algorithms need a validation process which consists of a 5-fold cross-validation made for each training fold independently.

The DCS methods that we have selected in our proposal, use K-NN algorithm to define the local region, and depending on the K value the results vary. Because of that we have run these methods over several K values: 6,12,18,24, 30 when K-NN is used and 3,6,9,12,15 when K-NNE is used. It is worth mentioning that as in each sub-problem there are two classes and K-NNE obtains the K nearest neighbors of each class, the number of neighbors that take part in K-NN and K-NNE are the same.

6.4. Obtained results

In this sub-section we show the results obtained by 4 different variations of DYNOVO.

Table 2 shows the results obtained by DYNOVO when K-NN is used to obtain the local regions. This table is separated into two sections: in the left side are shown the results obtained when OLA is used as DCS method (DYNOVO-OLA-KNN), whereas in the right side are shown those obtained when DW-OLA is used (DYNOVO-DW-OLA-KNN).

Table 3 shows the results obtained by DYNOVO when K-NNE is used to obtain the local regions. As in the previous table, this time also the table is divided into two parts: in the left side are shown the results obtained when OLA is used as DCS method (DYNOVO-OLA-KNNE) and in the right side the results obtained when DW-OLA is used (DYNOVO-DW-OLA-KNNE).

For each DYNOVO variation the average of the best K is remarked in bold. These K values are used in the next sub-section to compare DYNOVO's variations with other state of the art methods.

	DYNOVO-OLA-KNN					DYNOVO-DW-OLA-KNN				
DB	K=6	K=12	K=18	K=24	K=30	K=6	K=12	K=18	K=24	K=30
anneal	98.998	98.886	98.886	98.664	98.775	99.220	99.109	99.332	99.220	99.109
balance-scale	89.600	89.600	89.760	89.920	89.760	89.600	89.760	89.600	89.760	89.760
car	96.065	96.181	96.296	96.296	96.296	96.065	96.007	96.123	96.123	96.123
cmc	54.175	54.039	52.682	54.311	54.175	53.700	54.447	53.225	54.107	54.039
dermatology	96.721	97.268	97.541	97.541	97.541	96.721	96.721	96.448	97.268	97.268
ecoli	86.905	87.202	87.798	87.500	87.202	86.905	87.798	87.202	87.202	87.202
glass	68.224	71.495	70.093	70.093	69.626	68.692	71.495	70.561	71.028	71.028
iris	95.333	97.333	96.667	95.333	95.333	96.000	96.000	95.333	95.333	95.333
imgsegment	97.229	97.359	97.489	97.273	97.229	97.229	97.229	97.532	97.229	97.229
lymph	87.838	87.162	85.135	83.784	85.135	87.838	88.514	85.811	83.784	84.459
nursery	98.526	98.526	98.549	98.573	98.573	98.526	98.526	98.611	98.634	98.634
optdigits	98.310	98.203	98.132	98.132	98.132	98.238	98.149	98.043	98.043	98.060
page-blocks	97.278	97.058	97.077	97.040	97.131	97.223	97.150	97.223	97.186	97.278
pendigits	98.781	98.817	98.763	98.799	98.754	98.772	98.790	98.790	98.817	98.790
satimg	89.464	89.448	89.510	89.371	89.355	89.510	89.588	89.588	89.448	89.542
solar-flare1	70.279	70.279	69.969	69.969	69.969	71.827	71.827	71.517	71.207	71.827
solar-flare2	75.328	75.235	75.141	75.141	75.141	75.235	75.235	75.141	75.047	75.141
vehicle	73.995	74.823	74.586	74.232	74.586	73.404	74.941	73.759	73.759	74.941
vowel	90.909	89.192	89.495	89.495	89.293	91.818	90.909	90.808	90.909	90.909
waveform-5000	84.520	84.520	85.180	85.260	85.560	84.020	83.860	84.600	84.800	85.040
wine	95.506	95.506	95.506	95.506	95.506	96.629	96.629	96.629	96.629	96.629
zoo	97.030	97.030	97.030	97.030	97.030	97.030	97.030	97.030	97.030	97.030
Mean	88.228	**88.416**	88.240	88.148	88.186	88.373	**88.623**	88.314	88.298	88.426

Table 2: Classification accuracies of DYNOVO when K-NN is used to obtain the local region.

	DYNOVO-OLA-KNNE					DYNOVO-DW-OLA-KNNE				
DB	K=3	K=6	K=9	K=12	K=15	K=3	K=6	K=9	K=12	K=15
anneal	98.998	98.775	98.886	98.886	98.664	99.332	99.220	99.443	99.443	99.443
balance-scale	88.640	89.600	89.600	89.440	89.760	89.440	90.720	90.720	90.560	90.880
car	96.065	95.428	96.181	96.296	96.238	96.817	96.181	96.470	96.470	96.470
cmc	53.836	54.243	53.225	53.496	53.632	61.371	62.322	62.322	63.069	62.865
dermatology	96.995	97.541	96.995	97.268	97.541	96.995	96.995	96.721	97.268	97.541
ecoli	86.310	86.905	86.905	87.798	87.500	87.798	88.095	89.583	89.583	90.179
glass	72.897	71.495	71.963	71.495	71.028	75.701	76.636	75.234	75.701	75.234
iris	96.000	96.667	96.000	96.000	96.000	95.333	96.000	96.000	96.000	95.333
imgsegment	97.186	97.143	96.883	97.013	96.797	97.532	97.749	97.706	97.532	97.403
lymph	87.838	87.162	87.162	86.486	86.486	87.162	87.838	86.486	87.162	86.486
nursery	98.511	98.349	98.156	98.526	98.573	98.634	98.495	98.387	98.696	98.719
optdigits	98.149	98.132	98.096	98.025	97.954	98.096	98.096	98.060	98.007	97.883
page-blocks	96.638	96.821	96.675	96.656	96.620	97.625	97.625	97.533	97.552	97.460
pendigits	98.362	98.353	98.435	98.444	98.444	98.408	98.344	98.372	98.444	98.490
satimg	88.594	88.205	88.625	88.656	88.485	89.899	89.806	89.930	90.070	90.023
solar-flare1	70.279	69.969	69.969	69.659	69.659	73.994	73.994	73.994	73.684	73.375
solar-flare2	74.953	75.235	75.235	74.953	75.047	76.735	77.486	77.486	76.923	76.923
vehicle	74.823	74.704	73.641	75.296	75.414	80.733	82.151	82.388	83.097	83.688
vowel	90.404	90.000	89.293	89.293	88.889	91.818	91.616	90.909	90.707	91.616
waveform-5000	84.000	83.920	84.160	84.540	84.540	84.380	84.320	84.400	84.980	85.160
wine	95.506	95.506	95.506	95.506	95.506	96.067	96.629	96.629	96.629	96.629
zoo	97.030	97.030	97.030	97.030	97.030	97.030	97.030	97.030	97.030	97.030
Mean	**88.273**	88.236	88.119	88.217	88.173	89.586	89.879	89.809	89.937	**89.947**

Table 3: Classification accuracies of DYNOVO when K-NNE is used to obtain the local region.

6.5. Comparing the results

In this sub-section we compare our proposals with other state-of-the-art methods. We have divided the experiments into two parts: in the first one OLA is applied in those methods that select the classifiers dynamically, while in the second one DW-OLA is applied. We show the results of each part on Tables 4 and 5. Following, we briefly describe the strategies that correspond to each column of the tables.

- Best Single (OVO-BS) [12]: Each database is classified with every classifier defined in sub-section 6.2 applying OVO decomposition strategy. The result of the best base classifier is shown in each database.
- Galar et al. (Galar) [16]: It finds the K nearest neighbors of the test instance and it applies OVO only considering those classes in the neighborhood. The K value is established to 3 times the number of classes. The result of the best base classifier is shown in each database.
- Static selection (OVO-ST) [38]: For each sub-problem it is selected independently the base classifier that obtains the best result after a validation process.
- DCS methods (OLA [39] or DW-OLA [36]): Depending on the table the DCS strategy that is used vary. In the first table is OLA the strategy that is compared, while in the second table is DW-OLA. As it has been commented before, the DCS strategies are run over several K values, in the tables the results of the K value with the highest mean are shown.
- Dynamic selection of the base classifier in each sub-problem with K-NN (DYNOVO-OLA-KNN or DYNOVO-DW-OLA-KNN): It is tried to select the best base classifier in each sub-problem independently and dynamically. K-NN is used to obtain the local region in DCS strategies. The results obtained by the best K value in Table 2 are shown. In Figure 3(b) it can be seen a graphical example of how DYNOVO-OLA-KNN and DYNOVO-DW-OLA-KNN obtain the local regions.
- Dynamic selection of the base classifier in each sub-problem with K-NNE (DYNOVO-OLA-KNNE or DYNOVO-DW-OLA-KNNE): Similar to the previous one with the difference that it uses K-NNE instead of K-NN. The results obtained by the best K value in Table 3 are shown. In Figure 3(c) it can be seen a graphical example of how DYNOVO-OLA-KNNE and DYNOVO-DW-OLA-KNNE obtain the local regions.

Table 4 shows the results obtained when OLA is used in the methods that select dynamically the base classifiers. It could be seen that our proposal DYNOVO-OLA-KNN shows the best result in the majority of the cases: it reaches the best result in 8 of the databases. Moreover it achieves the best mean and rank values also. Our other proposal, DYNOVO-OLA-KNNE, obtains the best results in 4 of the databases and it gets the third best mean.

Table 5 shows the results obtained when DW-OLA is used in the methods that select dynamically the base classifiers. Our proposal DYNOVO-DW-OLA-KNNE shows the best result in 15 of the databases and it reaches also the best mean and rank. This time our other proposal, DYNOVO-DW-OLA-KNN, also gets interesting results since it obtains the second best mean and rank.

These results, show that methods which select the base classifiers dynamically in OVO obtain promising results. However, we can not obtain any meaningful conclusion without using a statistical test. Hence, in the next sub-section, we carry out an statistical analysis in order to find whether signicant differences among the results obtained exists or not.

6.6. Statistical analysis

As we have several methods to compare, according to García et al. [17], 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 signicant differences between them.

With respect to OLA the results of the statistical analysis reject the null hypothesis that all the methods are equivalent, since the p-value (0.0200) returned by the Iman-Davenport test is lower than our α-value (0.1). In Table 6

DB	OVO-BS	Galar	OVO-ST	OLA	DYNOVO-OLA-KNN	DYNOVO-OLA-KNNE
anneal	98.552	98.552	**98.998**	98.664	98.886	**98.998**
balance-scale	**90.400**	**90.400**	89.120	89.440	89.600	88.640
car	93.866	93.866	93.692	95.833	**96.181**	96.065
cmc	**54.582**	54.447	53.089	51.663	54.039	53.836
dermatology	97.541	**98.361**	96.995	95.902	97.268	96.995
glass	**73.832**	**73.832**	70.561	71.495	71.495	72.897
ecoli	86.607	86.905	85.417	86.607	**87.202**	86.310
imgsegment	97.186	97.013	97.143	97.143	**97.359**	97.186
iris	96.667	96.667	96.667	94.667	**97.333**	96.000
lymph	87.162	87.162	86.486	86.486	87.162	**87.838**
nursery	97.238	97.130	97.824	98.071	**98.526**	98.511
optdigits	98.292	**98.523**	98.256	98.256	98.203	98.149
page-blocks	**97.223**	97.168	97.003	97.003	97.058	96.638
pendigits	98.026	98.299	98.426	98.690	**98.817**	98.362
satimg	88.283	88.361	88.454	88.858	**89.448**	88.594
solar-flare1	70.279	70.588	69.969	**71.517**	70.279	70.279
solar-flare2	**75.516**	**75.516**	75.141	74.672	75.235	74.953
vehicle	75.414	75.887	**76.123**	74.941	74.823	74.823
vowel	82.828	83.636	84.949	83.232	89.192	**90.404**
waveform-5000	86.700	**86.720**	86.680	84.500	84.520	84.000
wine	**98.876**	**98.876**	96.067	98.315	95.506	95.506
zoo	96.040	96.040	95.050	95.050	**97.030**	**97.030**
Mean	88.232	88.361	87.823	87.773	**88.416**	88.273
Rank	3.16	2.93	4.20	4.16	**2.72**	3.82

Table 4: Classification accuracies of different methods. In those approaches that select the classifiers dynamically, OLA method is used.

we show the most relevant p-values obtained with Shaffer post-hoc test. Although there are not statistical differences in each pairwise comparisons, DYNOVO-OLA-KNN is close to outperform statistically OVO-ST and OLA, since the p-value is low. Because of that, and taking into account that the results obtained in Table 4, we consider that DYNOVO-OLA-KNN performs better than the other methods.

Considering DW-OLA, the Iman-Davenport test also returns p-value (0.0002) lower than α-value, so we execute the Shafer post-hoc test. The achieved p-values could be seen in Table 7. The results show that DYNOVO-DW-OLA-KNNE is the most robust strategy since it outperforms significantly OVO-BS, OVO-ST and DW-OLA.

6.7. Computational complexity

In order to provide a more complete study, we analyze the time and space complexity of our proposal.

The computational load of building the model is pretty big, since it involves to classify every training instance over every classifiers for every pair of classes. Those results are stored on a table, hence, this task only needs to be executed once. At classification time the information of the tables is retrieved from the table.

To analyse the computational and spatial complexity of classifying a new instance, let us examine the process that such instance undergoes.

- *For every pair of classes in the dataset, a vote is cast:* The number of pair of classes is $O(C^2)$, where C is the number of classes.
  - *Search the K nearest neighbors:* K-NN using kd-tree has a search time of $O(K\log(I_{TR}))$, where I_{TR} the number of instances in the training set from where the model has been built and K is the number of nearest neighbours.
  - *Search the classifier that best classifies the neighbors:* This is achieved by a search in a table that stores if a classifier type classified correctly an instance in the sub-problem associated to a pair of classes. The table has the pair of classes, the training instances and the classifier types as keys and a boolean as value. If implemented as a hash table, the searching time is $O(1)$ in the average.
  - *The instance is classified according to the best classifier:* It is clear that this depends of the classifier, but being K-NN a lazy algorithm, and thus a slow one, it looks sensible to assume $O(K\log I_{TR})$ is an upper bound in the execution time.
- *The instance is assigned the class with the majority of votes:* It takes $O(C^2)$ time to tally all the votes.

DB	OVO-BS	Galar	OVO-ST	DW-OLA	DYNOVO-DW-OLA-KNN	DYNOVO-DW-OLA-KNNE
anneal	98.552	98.552	98.998	99.220	99.109	**99.443**
balance-scale	90.400	90.400	89.120	89.120	89.760	**90.880**
car	93.866	93.866	93.692	95.833	96.007	**96.470**
cmc	54.582	54.447	53.089	51.663	54.447	**62.865**
dermatology	97.541	**98.361**	96.995	95.082	96.721	97.541
glass	73.832	73.832	70.561	71.495	71.495	**75.234**
ecoli	86.607	86.905	85.417	84.821	87.798	**90.179**
imgsegment	97.186	97.013	97.143	96.926	97.229	**97.403**
iris	**96.667**	**96.667**	**96.667**	94.667	96.000	95.333
lymph	87.162	87.162	86.486	**88.514**	**88.514**	86.486
nursery	97.238	97.130	97.824	98.071	98.526	**98.719**
optdigits	98.292	**98.523**	98.256	98.185	98.149	97.883
page-blocks	97.223	97.168	97.003	96.766	97.150	**97.460**
pendigits	98.026	98.299	98.426	98.717	**98.790**	98.490
satimg	88.283	88.361	88.454	88.827	89.588	**90.023**
solar-flare1	70.279	70.588	69.969	72.136	71.827	**73.375**
solar-flare2	75.516	75.516	75.141	74.578	75.235	**76.923**
vehicle	75.414	75.887	76.123	75.650	74.941	**83.688**
vowel	82.828	83.636	84.949	85.455	90.909	**91.616**
waveform-5000	86.700	**86.720**	86.680	83.920	83.860	85.160
wine	**98.876**	**98.876**	96.067	98.315	96.629	96.629
zoo	96.040	96.040	95.050	96.040	**97.030**	**97.030**
Mean	88.232	88.361	87.823	87.909	88.623	**89.947**
Rank	3.59	3.36	4.50	4.16	3.30	**2.09**

Table 5: Classification accuracies of different methods. In those approaches that select the classifiers dynamically, DW-OLA method is used.

Table 6: Shaffer test results when OLA is used

Hypothesis	p-value
DYNOVO-OLA-KNN vs OVO-ST	=(0.1323)
DYNOVO-OLA-KNNvs OLA	=(0.1323)
Galar vs OVO-ST	=(0.2405)
Galar vs OLA	=(0.2958)
DYNOVO-OLA-KNN vs DYNOVO-OLA-KNNE	=(0.5312)
OVO-BS vs OVO-ST	=(0.6383)
OVO-BS vs OLA	=(0.6383)
Galar vs DYNOVO-OLA-KNNE	=(0.8127)

Table 7: Shaffer test results when DW-OLA is used

Hypothesis	p-value
DYNOVO-DW-OLA-KNNE vs OVO-ST	**+(2.9E-4)**
DYNOVO-DW-OLA-KNNEvs DW-OLA	**+(0.0025)**
DYNOVO-DW-OLA-KNNE vs OVO-BS	**+(0.0783)**
DYNOVO-DW-OLA-KNNE vs Galar	=(0.2405)
DYNOVO-DW-OLA-KNNE vs DYNOVO-DW-OLA-KNN	=(0.3273)
DYNOVO-DW-OLA-KNN vs OVO-ST	=(0.3273)
Galar vs OVO-ST	=(0.3273)
OVO-BS vs OVO-ST	=(0.7493)
DYNOVO-DW-OLA-KNN vs OLA	=(0.8803)
Galar vs OLA	=(0.9509)

Within these assumptions the average execution time of all the process is $O(K\log(I_{TR})C^2)$. Let us note that I_{TR} is different for every pair of classes in the C^2 sub-problems, but in average will be $(N/C)*2$. If N is the number of instances in the original database, the average execution time will be $O(K\log(N)C^2)$, so the classification time is logarithmic in the number of instances in the original dataset and quadratic in the number of classes, provided reasonable classification complexity of the classifiers used.

With respect to space complexity, the approach would request $O(C^2 I_{TR} T)$ space, that, as stated above, amounts to $O(NCT)$, with N the number of instances in the original database. Storage of the classifier models should never be bigger than $O(KN)$, even with lazy paradigms using kd-trees structures or similar.

Compared with other OVO versions, our proposal has a bigger space complexity, due to the need of storing big tables. About time complexity, only the search for the K neighbours and the lookup in the hash table are not made in other OVO versions. As the comparisons with other methods are considerably in favour of DYNOVO-DW-OLA-

KNNE, and the differences with other OVO versions are mostly in space requirements, we consider that its good performance compensates this extra computational cost.

6.8. Discussion

After all these experiments, considering only the state-of-the-art methods, the first conclusion that we have obtained is that selecting different base classifier for each sub-problem statically in OVO (OVO-ST), does not outperform the best single classifier in OVO (OVO-BS). These results coincide with those found in the state-of-the-art [27] [37]. On the other hand, it is worth mentioning that the algorithm proposed by Galar et al. [16] obtains interesting result and although it uses less sub-problems than OVO, it shows the best performance among the state-of-the-art methods.

On the other hand it can be seen that the proposed approach obtains promising results. It gets better mean and rank than the compared methods with almost all the variations (the exception is DYNOVO-OLA-KNNE strategy). Moreover the statistical tests show the good performance of our proposal.

Finally DYNOVO-DW-OLA-KNNE is which shows the best performance. It obtains the best mean with a significant difference and the statistical test shows its solidity. Furthermore, it can be seen in Table 3 that all the averages obtained with the different K values overcome the averages obtained by state-of-the-art methods. The combination of DW-OLA and K-NNE gives some advantages which result beneficial to select the appropriate base classifier. Let us consider that we are trying to select the appropriate base classifier to classify a new unknown instance for the sub-problem that distinguishes between θ_i and θ_j classes. Also consider that all its K nearest nehighbors belong to θ_i class. Under these circumstances, it is more likely to select a base classifier that tends to return θ_i class. But if the new unknown sample belongs to θ_j, it is more likely to predict the wrong class. This problem can be minimized using K-NNE algorithm, since it gives the chance to participate to all the classes. In this manner the selected base classifier should be able to differentiate both classes. However, it is possible to be a significant difference in the distance to the new unknown sample between the K nearest neighbors of θ_i and θ_j. Therefore it is not completely adequate that all the neighbors have the same influence when the base classifier is selected. So one possibility is to assign different weights to each neighbor depending in their distance to the new sample, in other words, apply DW-OLA.

7. Conclusion

In this paper we present a new proposal called DYNOVO which aims to improve classification accuracy in supervised classification multi-class problems. Among several base classifiers, the approach attempts to select the best base classifier in OVO dynamically for each test patterns. To do so we have chosen several well-known classifiers from different Machine Learning paradigms: SVM, C4.5 Decision Tree, Ripper, Bayes Networks and Naive Bayes. We have presented 4 different variations of our proposal which have been tested over 22 databases from the UCI repository.

The novel procedure proposed has shown its usefulness due to the competitive results obtained. We have shown the positive synergy existing between OVO and DCS strategies, specially when K-NNE is utilized to obtain the local region and the instances of the local region are weighted by the distance to the new unknown case.

This fact open doors for future combinations of OVO and DCS using more complex DCS strategies or to extend it to Dynamic Ensemble Selection strategies. Furthermore, it would be interesting to introduce these strategies in the more general ECOC framework.

Acknowledgements

The work described in this paper was partially conducted within the Basque Government Research Team Grant IT313-10 and the University of the Basque Country UPV/EHU. I. Mendialdua holds a Grant from Basque Government.

References

[1] Aha, D.W., Kibler, D., Albert, M.K., 1991. Instance-based learning algorithms. Machine learning 6, 37–66.
[2] Anand, R., Mehrotra, K., Mohan, C.K., Ranka, S., 1995. Efficient classification for multiclass problems using modular neural networks. IEEE Transactions on Neural Networks 6, 117–124.

[3] Arruti, A., Mendialdua, I., Sierra, B., Lazkano, E., Jauregi, E., 2014. New one versus allone method: Nov@. Expert Systems with Applications 41, 6251–6260.
[4] Bache, K., Lichman, M., 2013. UCI machine learning repository. URL: `http://archive.ics.uci.edu/ml`.
[5] 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.
[6] 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.
[7] Cavalin, P.R., Sabourin, R., Suen, C.Y., 2013. Dynamic selection approaches for multiple classifier systems. Neural Computing and Applications 22, 673–688.
[8] Cohen, W.W., 1995. Fast effective rule induction, in: Proceedings of the Twelfth international conference on machine learning, pp. 115–123.
[9] Dietterich, T.G., Bakiri, G., 1995. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263–286.
[10] Dos Santos, E.M., Sabourin, R., Maupin, P., 2008. A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. Pattern Recognition 41, 2993–3009.
[11] Fei, B., Liu, J., 2006. Binary tree of svm: a new fast multiclass training and classification algorithm. IEEE Transactions on Neural Networks 17, 696–704.
[12] Friedman, J., 1996. Another approach to polychotomous classifcation. Technical Report. Technical report, Stanford University, Department of Statistics.
[13] Friedman, N., Geiger, D., Goldszmidt, M., 1997. Bayesian network classifiers. Machine learning 29, 131–163.
[14] Fürnkranz, J., 2002. Round robin classification. The Journal of Machine Learning Research 2, 721–747.
[15] 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.
[16] 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.
[17] 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.
[18] García-Pedrajas, N., Ortiz-Boyer, D., 2011. An empirical study of binary classifier fusion methods for multiclass classification. Information Fusion 12, 111–130.
[19] Giacinto, G., Roli, F., 1999. Methods for dynamic classifier selection, in: International Conference on Image Analysis and Processing, IEEE Computer Society. pp. 659–659.
[20] 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.
[21] Hastie, T., Tibshirani, R., 1998. Classification by pairwise coupling. The annals of statistics 26, 451–471.
[22] 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.
[23] John, G.H., Langley, P., 1995. Estimating continuous distributions in bayesian classifiers, in: Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc.. pp. 338–345.
[24] Kapp, M.N., Sabourin, R., Maupin, P., 2012. A dynamic model selection strategy for support vector machine classifiers. Applied Soft Computing 12, 2550–2565.
[25] Ko, A.H., Sabourin, R., Britto Jr, A.S., 2008. From dynamic classifier selection to dynamic ensemble selection. Pattern Recognition 41, 1718–1731.
[26] Lebrun, G., Lezoray, O., Charrier, C., Cardot, H., 2007. An ea multi-model selection for svm multiclass schemes, in: Proceedings of the Ninth international work conference on Artificial neural networks (IWANN07), Springer. pp. 260–267.
[27] Liepert, M., 2003. Topological fields chunking for german with svms: Optimizing svm-parameters with gas, in: Proceedings of the International Conference on Recent Advances in Natural Language Processing.
[28] Liu, R., Yuan, B., 2001. Multiple classifiers combination by clustering and selection. Information Fusion 2, 163–168.
[29] Lorena, A.C., de Carvalho, A.C., Gama, J.M., 2008. A review on the combination of binary classifiers in multiclass problems. Artificial Intelligence Review 30, 19–37.
[30] Lorena, A.C., De Carvalho, A.C., 2008. Evolutionary tuning of svm parameter values in multiclass problems. Neurocomputing 71, 3326–3334.
[31] 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.
[32] Platt, J.C., Cristianini, N., Shawe-Taylor, J., 2000. Large margin dags for multiclass classification. Advances in neural information processing systems 12, 547–553.
[33] Quinlan, J.R., 1993. C4. 5: programs for machine learning. volume 1. Morgan kaufmann.
[34] Reid, S.R., 2010. Model Combination in Multiclass Classification. Ph.D. thesis. University of Colorado.
[35] Sierra, B., Lazkano, E., Irigoien, I., Jauregi, E., Mendialdua, I., 2011. K nearest neighbor equality: Giving equal chance to all existing classes. Information Sciences 181, 5158–5168.
[36] Smits, P.C., 2002. Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection. IEEE Transactions on Geoscience and Remote Sensing 40, 801–813.
[37] de Souza, B.F., de Carvalho, A., Calvo, R., Ishii, R.P., 2006. Multiclass svm model selection using particle swarm optimization, in: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems, IEEE. pp. 31–31.
[38] Szepannek, G., Bischl, B., Weihs, C., 2009. On the combination of locally optimal pairwise classifiers. Engineering Applications of Artificial Intelligence 22, 79–85.
[39] Woods, K., 1997. Combination of multiple classifiers using local accuracy estimates. IEEE Transactions on Pattern Analysis and Machine

Intelligence 19, 405–410.