Pattern Recognition Letters 31 (2010) 2375–2388

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Analysis of an evolutionary RBFN design algorithm, CO^{2}RBFN, for imbalanced data sets

María Dolores Pérez-Godoy *, Alberto Fernández, Antonio Jesús Rivera, María José del Jesus

Department of Computer Science, University of Jaén, Jaén, Spain

ARTICLE INFO

Article history:
Received 8 September 2009
Available online 23 July 2010
Communicated by E. Bernado-Mansilla

Keywords:
Neural networks
Radial-basis function networks
Genetic algorithm
Imbalanced data sets
SMOTE pre-processing method

ABSTRACT

In the classification problem field, we often encounter many real application areas in which the data do not have an equitable distribution among the different classes of the problem. In such cases, we are dealing with the so-called imbalanced data sets. This scenario has significant interest since standard classifiers are often biased towards the majority classes, whereas the minority ones tend to have a higher reward as they usually define the concepts of interest from the learning point of view.

The aim of this paper is to analyse the performance of CO^{2}RBFN, a evolutionary cooperative–competitive model for the design of radial-basis function networks applied to classification problems on imbalanced domains, and to study its cooperation with a well-known pre-processing method, the "synthetic minority over-sampling technique". The good performance of CO^{2}RBFN is shown through an experimental study carried out on a large collection of imbalanced data sets where we compare, by means of a proper statistical study, the behaviour of our model with many representative neural networks algorithms, the C4.5 decision tree and a hierarchical fuzzy rule-based classification system.

1. Introduction

Radial-basis function networks (RBFNs) are one of the most important artificial neural network paradigms in the field of machine learning. An RBFN is a feed-forward artificial neural network with a single layer of hidden units, called radial-basis functions (RBFs) (Broomhead and Lowe, 1988; Moody and Darken, 1989). RBFNs have important features such as: a simple topological structure; the possibility of extracting rules (Jang and Sun, 1993; Jin and Sendhoff, 2003); a universal approximation capability (Park and Sandberg, 1991; Park and Sandberg, 1993); and each neuron/RBF has a characteristic locally-tuned response that depends on its centre and its width (radius).

One important paradigm for RBFN design is evolutionary computation (Holland, 1975; Goldberg, 1989; Bäck et al., 1997), which uses natural evolution and stochastic search to design optimization algorithms. More precisely, evolutionary computation maintains a population of individuals which evolves according to the operators as mutation, recombination or selection, and each individual in the population receives a measure of its fitness in the environment.

CO^{2}RBFN (Pérez-Godoy et al., 2010) is a hybrid cooperative–competitive evolutionary method for the design of RBFNs. In this environment an RBF represents an individual and, therefore, each RBF competes for survival and all the RBFs in the population cooperate towards a definite solution. The key matter of this model is the computation of the credit assignment (fitness) of each individual. In this manner, three parameters have been defined in order to measure the role of an RBF: its contribution to the output of the network, the error and the overlapping of the neuron. The decision for the application of the different evolutionary operators to the RBFs is defined using a fuzzy rule base system.

The overall efficiency of RBFNs has been proved in many areas like pattern classification (Buchtala et al., 2005), function approximation (Park and Sandberg, 1991), time-series prediction (Whitehead and Choate, 1996) and multiple specific applications such as credit assessment (Lacerda et al., 2005), face recognition (Er et al., 2005), image recognition (Siddiqui et al., 2009), process control (Huang et al., 2008), medical diagnosis (Maglogiannis et al., 2008; Marcos et al., 2008), financial time-series forecasting (Sun et al., 2005) and intrusion detection (Zhang et al., 2005), among others. In most of these areas, the data sets have a common and usual characteristic: they are imbalanced data sets (Chawla et al., 2004).

The problem of imbalanced data sets in classification (Chawla et al., 2004) occurs when the number of instances of one class is much lower than the instances of the other class(es). Since standard learning algorithms are developed to maximise the standard accuracy rate, which is independent of the class distribution, in this context this causes a bias towards the majority class in the training

* Corresponding author. Tel.: +34 953 212891; fax: +34 953 212472.
E-mail addresses: lperez@ujaen.es (M.D. Pérez-Godoy), alberto.fernandez@ujaen.es (A. Fernández), arivera@ujaen.es (A.J. Rivera), mjjesus@ujaen.es (M.J. del Jesus).

0167-8655/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2010.07.010

of classifiers and results in a lower sensitivity in detecting the minority class examples.

A large number of approaches have been previously proposed to deal with this problem, which can be categorised into two groups: the internal approaches which create new algorithms or modify existing ones to take the class-imbalance problem into consideration (Barandela et al., 2003; Xu et al., 2007) and external approaches which pre-process the data in order to diminish the effect caused by their class imbalance (Batista et al., 2004; Estabrooks et al., 2004). The internal approaches have the disadvantage of being algorithm specific, whereas external approaches are independent of the classifier used and are, for this reason, more versatile.

In this paper, CO^{2}RBFN is applied to solving imbalanced classification problems (data sets obtained from the UCI repository (Asuncion and Newman, 2007)). According to the previous fact, we study the effect of a pre-processing stage (applying the "synthetic minority over-sampling technique" (SMOTE) Chawla et al., 2002) in the performance of CO^{2}RBFN by contrasting the results obtained using the original data sets. Furthermore, we extend our experimental study with some well-known classification models for comparison, including many artificial neural networks algorithms, the C4.5 decision tree (Quilan, 1993) and a hierarchical fuzzy rule-based classification system (HFRBCS) (Fernández et al., 2009). Furthermore, our experimental results are supported by means of a strong statistical study using non-parametric tests as stated in (Demšar, 2006; García and Herrera, 2008; García et al., 2009).

In order to do this, this paper is arranged as follows: in Section 2 a review of the RBFN design is detailed. In Section 3 we introduce the problem of imbalanced data sets, describing its features, the SMOTE pre-processing technique used to deal with this problem, the specific evaluation metrics for this scenario and a revision of the application of the RBFN to the imbalanced problem. Then, Section 4 presents in detail the CO^{2}RBFN approach. Next, in Section 5 we introduce the experimental framework with the collection of data sets used in the empirical study and all the parameters for CO^{2}RBFN and the different algorithms of comparison. All the experimental results and the analysis of the behaviour of CO^{2}RBFN are presented in Section 6, whereas the concluding remarks and future work are presented in Section 7. Finally, we include an appendix with the complete tables of results and a brief description of the statistical tests used for the comparative study.

2. Design of RBFNs

An RBFN is a feed-forward neural network with three layers: an input layer with n nodes, a hidden layer with m neurons or RBFs, and an output layer with one or several nodes (Fig. 1).

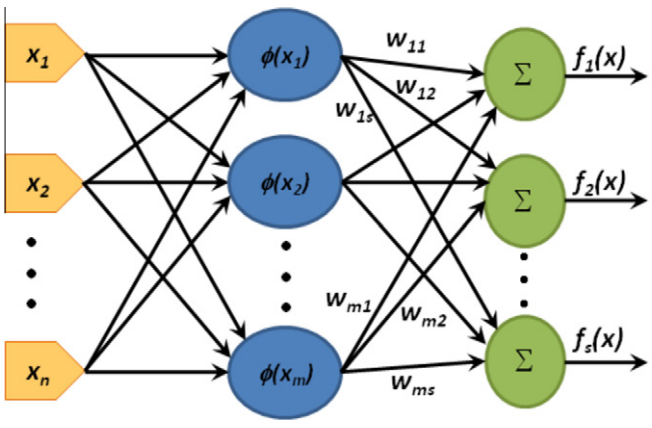

Fig. 1. RBFN Topology of RBFN.

Each input node corresponds to a feature of the input pattern. The m neurons of the hidden layer are activated by a radially-symmetric basis function, ϕ_i: $R^n \to R$, which can be defined in several ways (Rojas et al., 1997). The Gaussian function is the most widely used: $\phi_i(\vec{x}) = \phi_i(e^{-(\|\vec{x}-\vec{c_i}\|/d_i)^2})$, where $\vec{c_i} \in R^n$ is the centre of basis function ϕ_i, $d_i \in R$ is the width (radius), and $\|\|$ is typically the Euclidean norm on R^n. This expression is the one used in this paper as the RBF. The output nodes implement the function in Eq. (1):

$$f(\vec{x}) = \sum_{i=1}^{m} w_{ij}\phi_i(\vec{x}) \tag{1}$$

The output of one basis function will be high when the input vector and the centre of this basis function are closer, always taking into account the value of the radius. The weights w_{ij} show the contribution of an RBF to the respective output node, and therefore the output nodes implement the weighted sum of RBF outputs.

The traditional RBFN learning procedure has two stages: first, unsupervised learning of centres and widths is used, and finally output weights are established by means of supervised learning. Clustering techniques (Pedrycz, 1998) are normally used to adjust the centres. Regarding the widths, they may all be given the same value, may reflect the width of the previously calculated clusters (i.e. RBFs), or may be established as the average distance between RBFs, among other possibilities. In order to obtain the weights in the second stage, algorithms such as least mean square (LMS) (Widrow and Lehr, 1990) or singular value decomposition (widely known as SVD) (Golub and Van Loan, 1996) can be used.

As well as this typical methodology, different design strategies for RBFN design can be found in the literature. Due to the fact that RBFNs were initially used for function approximation, most of the methods are based on traditional optimization techniques such as regularization (Orr, 1995), orthogonalization of regressors (Chen et al., 1991), gradient-based (Neruda and Kudová, 2005), or Levenberg–Marquardt (Ampazis and Perantonis, 2002). These techniques can be used to decide the RBFs to aggregate or eliminate and may be considered as forward or backward selection methods (Peng et al., 2006).

Another important paradigm for RBFN design is evolutionary computation (Holland, 1975; Goldberg, 1989; Bäck et al., 1997), a general stochastic optimization framework inspired by natural evolution. In any evolutionary algorithm, and therefore in those for the evolutionary design of RBFNs, two main aspects must be considered: what is codified in an individual and the way to compute the goodness of this individual.

Regarding the first aspect, in the specialised literature most of the evolutionary proposals for the design of RBFNs (Harpham et al., 2004; Rivas et al., 2004; Lacerda et al., 2005) codify a complete RBFN by means of an individual, and the population of RBFNs evolves through different operators. This is known as the Pittsburgh representation scheme. In this paradigm the fitness of an individual usually is the classification rate of the model codified.

Nevertheless, according to Potter and De Jong (2000) evolutionary computation has some difficulties in solving certain types of problems, especially when an individual represents a complete solution composed of independent subcomponents. An alternative to the classical (Pittsburgh) approach is the cooperative–competitive evolutionary strategy (Whitehead and Choate, 1996; Potter and De Jong, 2000), which provides a framework where an individual of the population represents only a part of the solution, evolving in parallel, competing to survive but at the same time cooperating in order to find a common solution (the complete RBFN).

Regarding this approach, two main problems must be addressed:

1. The credit assignment, or the fitness allocated to each individual according to its contribution to the final solution.
2. The mechanism used in order to maintain diversity among individuals of the population.

The cooperative–competitive approach reduces the search space for the GA. The representation of the solution with an adequate credit assignment function is crucial to obtain an RBFN composed of a small number of accurate RBFs which do not overlap and which represent the information of the data examples.

In the literature there are some proposals concerning the design of RBFNs based on cooperative–competitive evolutionary strategies (Whitehead and Choate, 1996; Topchy et al., 1997; Rivera et al., 2007; Li et al., 2008). The most traditional one has been proposed by Whitehead and Choate (1996) where an individual represents an RBF and the population the whole network. Individual credit assignment is defined depending on the weight of the RBF. In the same way, in the algorithm described in (Topchy et al., 1997) an individual is an RBF and credit assignment is calculated according to the efficiency of the RBF or its contribution to the correct output of the network. In (Rivera et al., 2007) an individual represents an RBF and the method uses a credit assignment function witch combines concepts such as cooperation, specialization and niching. In (Li et al., 2008), a co-evolutionary RBFN design method is proposed, where interacting co-adapted subpopulations evolve independently. Each individual in a population represents a particular component (group of RBFs) of the RBFN. The fitness of an individual from a particular subpopulation is assessed by associating it with representatives from other subpopulations.

In this paper we make use of the CO^{2}RBFN algorithm, which is also a cooperative–competitive method for the design of RBFNs that has been developed by the authors and applied to standard classification problems in (Pérez-Godoy et al. (2010)).

3. Imbalanced data sets in classification

In this section, we first introduce the problem of imbalanced data sets. Then, we describe the pre-processing technique we have applied in order to deal with the imbalanced data sets: the SMOTE algorithm (Chawla et al., 2002). Afterwards, we will present the evaluation metrics for this type of classification problem. Finally a revision of RBFNs in imbalanced problems is shown.

3.1. The problem of imbalanced data sets

In some classification problems, the number of instances of every class is different. Particularly, in the binary case, the class-imbalance problem occurs when one class is represented by a large number of examples, whereas the other is represented by only a few (Chawla et al., 2004; Sun et al., 2009; He and Garcia, 2009).

The problem of imbalanced data sets is extremely significant (Yang and Wu, 2006) because it is implicit in most real world applications, such as satellite image classification (Suresh et al., 2008), risk management (Huang et al., 2006), optical remote-sensing data (Williams et al., 2009) and especially in medical applications (Kilic et al., 2007; Mazurowski et al., 2008; Peng and King, 2008). It is important to remark that usually the minority class represents the concept of interest, for example patients with illness in a medical diagnosis problem; whereas the other class represents the counterpart of that concept (healthy patients).

Standard classifier algorithms usually have a bias towards the majority class, since the rules which predict the higher number of examples are positively weighted during the learning process in favour of the standard accuracy rate metric, which does not take into account the class distribution of the data. Consequently, the instances belonging to the minority class are misclassified more often than those belonging to the majority class.

Another important issue of this problem are the small disjuncts that can be found in the data set (Weiss and Provost, 2003), which are regions of few examples of one class surrounded by many examples from the opposite class, and the difficulty of most learning algorithms in detecting these areas (Orriols-Puig et al., 2009; Orriols-Puig and Bernadó-Mansilla, 2009). In fact, learning algorithms try to benefit those models with a higher degree of coverage and these small disjuncts imply the application of very specific models which are discarded in favour or more general ones.

Furthermore, another handicap of imbalanced data sets, which is related to the presence of small disjuncts, is the overlapping between the examples of the positive and the negative class (García et al., 2008), in which the minority class examples can be simply treated as noise and ignored by the learning algorithm. These phenomena are depicted in Fig. 2(a) and (b) respectively.

As we stated in the introduction of the paper, a large number of approaches have previously been proposed for dealing with the class-imbalance problem. These approaches can be categorised into two groups: the internal approaches which create new algorithms or modify existing ones to take the class-imbalance problem into consideration (Barandela et al., 2003; Wu and Chang, 2005; Xu et al., 2007) and the external approaches which pre-process the data in order to diminish the effects of their class imbalance (Batista et al., 2004; Estabrooks et al., 2004). Furthermore, cost-sensitive learning solutions incorporating both the data and algorithmic level approaches assume higher misclassification costs with samples in the minority class and seek to minimise the high cost errors (Domingos, 1999; Zhou and Liu, 2006; Sun et al., 2007).

The great advantage of the external approaches is that they are more versatile, since their use is independent of the classifier selected. Furthermore, we may pre-process all data sets beforehand in order to use them to train different classifiers. In this manner, the computation time needed to prepare the data is lower.

3.2. Pre-processing imbalanced data sets. The SMOTE algorithm

As mentioned before, applying a pre-processing step in order to balance the class distribution is a positive solution to the imbalanced data set problem (Batista et al., 2004). Specifically, in this work we have chosen an over-sampling method which is widely-used in this area: the SMOTE algorithm (Chawla et al., 2002).

With this approach, the positive class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbours. Depending upon the amount of over-sampling

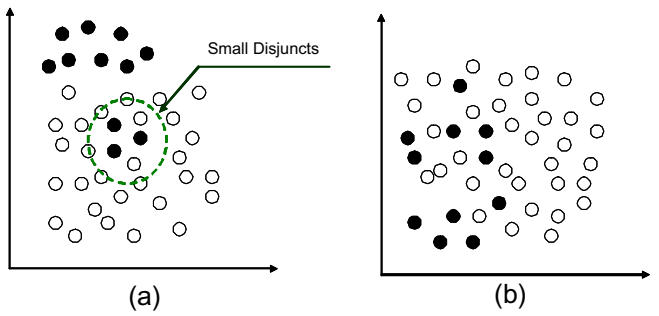

Fig. 2. Example of the imbalance between classes: (a) small disjuncts, (b) overlapping between classes.

required, neighbours from the k nearest neighbours are randomly chosen. This process is illustrated in Fig. 3, where x_i is the selected point, x_{i1} to x_{i4} are some selected nearest neighbours and r_1 to r_4 the synthetic data points created by the randomised interpolation. The implementation of this work uses only one nearest neighbour with the Euclidean distance, and balances both classes to 50% distribution.

Synthetic samples are generated in the following way: take the difference between the feature vector (sample) under consideration and its nearest neighbour. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general. An example is detailed in Fig. 4.

In short, the main idea is to form new minority class examples by interpolating between several minority class examples that lie together. In contrast with the common replication techniques (for example random over-sampling), in which the decision region usually become more specific, with SMOTE the overfitting problem is somehow avoided by causing the decision boundaries for the minority class to be larger and to spread further into the majority class space, since it provides related minority class samples to learn from. Specifically, selecting a small *k*-value could also avoid the risk of including some noise in the data.

3.3. Evaluation in imbalanced domains

The measures of the quality of classification are defined from a confusion matrix (shown in Table 1) which records correctly and incorrectly recognised examples for each class.

The most used empirical measure, accuracy rate (2), does not distinguish between the number of correct labels of different classes, which in the context of imbalanced problems may lead to erroneous conclusions. For example a classifier that obtains an accuracy of 99% in a data set with a distribution of 1:100, might not be accurate if it does not cover correctly any minority class instance:

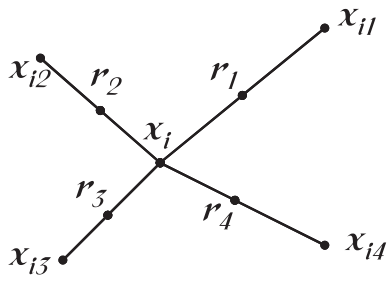

Fig. 3. An illustration of how to create the synthetic data points in the SMOTE algorithm.

Table 1
Confusion matrix for a two-class problem.

	Positive prediction	Negative prediction
Positive class	True positive (TP)	False negative (FN)
Negative class	False positive (FP)	True negative (TN)

$$\mathrm{Acc} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FN}+\mathrm{FP}+\mathrm{TN}}. \tag{2}$$

Because of this, instead of using accuracy, better suited metrics are considered. Two common measures, sensitivity and specificity (Eqs. (3) and (4)), approximate the probability of the positive (negative) label being true. In other words, they assess the effectiveness of the algorithm on a single class:

$$\text{sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \tag{3}$$

$$\text{specificity} = \frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}}. \tag{4}$$

The metric used in this work is the geometric mean of the true rates (Barandela et al., 2003), which can be defined as

$$\mathrm{GM} = \sqrt{\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \cdot \frac{\mathrm{TN}}{\mathrm{FP}+\mathrm{TN}}}. \tag{5}$$

This metric attempts to maximise the accuracy of each one of the two classes with a good balance. It is a performance metric that links both objectives.

3.4. RBFNs in imbalanced problems

As we stated before, there are both internal and external solutions to the problem of imbalanced data sets in classification. Specifically, in the framework of RBFNs we can some find examples of these approaches.

Regarding the internal approaches, they can include the use of a cost function in the training process to compensate class imbalance, and strategies to reduce the impact of the cost function in data probability distribution (Alejo et al., 2007).

There are also external approaches for solving this problem. In (Al-Haddad et al., 2000) different studies have been done to assess the effect of the size of the training data set on the accuracy for a microalgae problem. In (Murhphey and Guo, 2004) the random over-sampling method and snowball training are used as pre-processing method and the results are evaluated with three different artificial neural network architectures: multilayer back propagation network, RBFNs and Fuzzy ARTMAP. In (Padmaja et al., 2007), the over-sampling method SMOTE, has been used with RBFNs showing good behaviour. In (Li et al., 2006) an undersampling method is used to remove some training majority class patterns, before applying the RBFN. Padmaja et al. (2008) proposes a majority-filter based minority prediction (MFMP) unsupervised

```
Consider a sample (6,4) and let (4,3) be its nearest neighbour.
(6,4) is the sample for which k-nearest neighbours are being
       identified (4,3) is one of its k-nearest neighbours.
Let:  f1_1 = 6 f2_1 = 4, f2_1 - f1_1 = -2
       f1_2 = 4 f2_2 = 3, f2_2 - f1_2 = -1.
The new samples will be generated as
       (f1',f2') = (6,4) + rand(0-1) * (-2,-1)
       rand(0-1) generates a random number between 0 and 1.
```

Fig. 4. Example of the SMOTE application.

approach for selecting samples for supervised learners, such as: decision trees, *k*-nearest neighbour, Naïve Bayes and RBFNs.

Wu and Chow (2004), Rui and Minghu (2008) are examples of imbalanced problems where different neural networks models, RBFNs included, are used without a pre-processing phase. In (Zhao, 2009) a novel modular neural network is proposed to solve multi-class problems with imbalanced training sets, in which the results are compared with a RBFN model.

4. CO²RBFN for imbalanced data sets

CO²RBFN, is an evolutionary cooperative–competitive hybrid algorithm whose objective is to design simple and accurate RBFNs. In this paper the algorithm is applied to imbalanced data set classification.

A simple RBFN is composed of a low number of RBFs, which represent accurately the knowledge about the patterns of their environment and which are correctly located in the space of patterns (with minimum overlapping). Also, the RBFs must work well together in order to obtain an RBFN with adequate generalization.

In order to obtain simple and accurate RBFNs, CO²RBFN follows the evolutionary cooperative–competitive strategy, where each individual of the population represents an RBF (Gaussian function will be considered as RBF) and the entire population is responsible for the definite solution. This paradigm provides a framework where an individual of the population represents only a part of the solution, competing to survive (since it will be eliminated if its performance is poor) but at the same time cooperating in order to build the whole RBFN, which adequately represents the knowledge about the problem and achieves good generalization for new patterns. In this scenario, the local operation (RBFs with local response) and the representation of the majority of the examples (by means of any RBF) is reinforced, and the overlapping among RBFs is minimised. These design guidelines of CO²RBFN improve the interpretability of the RBFN obtained.

This evolutionary cooperative–competitive paradigm is reinforced with the remaining design components: fitness function, evolutionary operators and the fuzzy rule base system (FRBS), which decides the probability of applying operators.

In this environment, in which the final solution depends on the behaviour of many components, the fitness of each individual is known as credit assignment. In order to measure the credit assignment of an individual, three factors have been used to evaluate the role of each RBF in the network (error, contribution and overlapping). These factors reinforce: the individual quality of RBFs (calculating the local error inside the radius of each RBF), their generality (measured by the contribution which promotes RBFs with a high number of patterns inside its radius), and the adequate location of the RBFs in the space of patterns (measured by the overlapping). Together these three factors enhance the individual role of RBFs and their cooperative work toward building an accurate and simple network. It can be highlighted that the last two factors take into account all the patterns and the behaviour of the rest of the RBFs, and this can improve the efficiency of the algorithm in an imbalanced classification scenario. In this way, the fitness function of CO²RBFN combines concepts such as cooperation, specialization and niching (Buchtala et al., 2005).

There are four evolutionary operators that can be applied to an RBF: an operator that eliminates the RBF, two operators that mutate the RBF, and finally an operator that maintains the RBF parameters in order to explore and exploit the search space and to preserve the best RBF, respectively.

The application of the operators is determined by a FRBS. The inputs of this system are the three parameters used for credit assignment and the outputs are the operators' application probability. To design the set of rules we must take into account the fact that an RBF is worse if its contribution is low, its error is high and its overlapping is also high, otherwise it is better. In this way the probability of eliminating an RBF is high when this RBF is worse and so on.

The main steps of CO²RBFN, explained in the following subsections, are shown in the pseudocode in Fig. 5.

4.1. RBFN initialization

To define the initial network, with a number of RBFs established by the size of the population, a simple process is used: a specified number, m, of neurons (i.e. the size of population) is randomly allocated among the different classes of the training set. To do this, each RBF centre, $\vec{c}_i$, is randomly established to a pattern of the training set, taking into account that the RBFs must be distributed equally among the different classes. The RBF widths, d_i, will be set to half the average distance between the centres. Finally, the RBF weights, w_{ij}, are set to zero.

4.2. RBFN training

During this stage, RBF weights are trained. The LMS algorithm (Widrow and Lehr, 1990) has been used to calculate the RBF weights. This technique exploits the local information that can be obtained from the behaviour of the RBFs. The Eq. (6) shows the update of the weights:

$$\bar{w}_{k+1} = \bar{w}_k + \alpha \frac{e_k \bar{x}_k}{|\bar{x}_k^2|}, \tag{6}$$

where k is the number of iteration, $\bar{w}_{k+1}$ is the next value of the weight vector, $\bar{w}_k$ is the present value of the weight vector and $\bar{x}_k$ is the value of the actual input pattern vector. The present linear error, e_k, is defined as the difference between the desired output and the output network before adaptation. The α value is the *speed of learning*, it measures the size of the adjustment to be made. The choice of α controls stability and speed of convergence.

4.3. RBF evaluation

A credit assignment mechanism is required in order to evaluate the role of each basis function in the cooperative–competitive environment.

For an RBF ϕ_i, three parameters, a_i, e_i and o_i are defined:

- The contribution, a_i, of the RBF ϕ_i, $i = 1,\ldots,m$, is determined by considering the weight, w_i, and the number of patterns of the training set inside its width, npi_i. An RBF with a low weight and few patterns inside its width will have a low contribution:

$$a_i = \begin{cases} |w_i| & \text{if} \quad npi_i > q \\ |w_i| * (npi_i/q) & \text{otherwise} \end{cases} \tag{7}$$

  where q is the average of the npi_i values minus the standard deviation of the npi_i values.

```
Inicialise RBFN
While (Not Stop condition) Do
     Train RBFN
     Evaluate RBFs
     Apply operators to RBFs
     Substitute the eliminated RBFs
     Select the best RBFs
End_While
```

Fig. 5. Main steps of CO²RBFN.

- The error measure, e_i, for each RBF ϕ_i, is obtained by counting the wrongly classified patterns inside its radius:

$$e_i = \frac{npibc_i}{npi_i}, \tag{8}$$

where $npibc_i$ and npi_i are the number of wrongly classified patterns and the number of all patterns inside the RBF width respectively. It must be noted that this error measure does not consider imbalance among classes.
- The overlapping of the RBF ϕ_i and the other RBFs is quantified by using the parameter o_i. This parameter is calculated by taking into account the fitness sharing (Goldberg, 1989) methodology, whose aim is to maintain diversity in the population. This factor is expressed as

$$o_i = \sum_{j=1}^{m} o_{ij} \quad o_{ij} = \begin{cases} (1 - \|\phi_i - \phi_j\|/d_i) & \text{if} \quad \|\phi_i - \phi_j\| < d_i \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where o_{ij} measures the overlapping of the RBF ϕ_i and ϕ_j $j = 1,\ldots,m$.

4.4. Applying operators to RBFs

In this algorithm four operators have been defined:

- Operator *Remove*: eliminates an RBF.
- Operator *Random Mutation*: modifies the centre and width of an RBF. The width is altered with a probability inversely proportional to the number of features of the classification problem (n), in a percentage below 50% of the old width. The coordinates of the centre are modified as follows: if the coordinate is a real value, it is increased or decreased in a percentage below 50% of the width. If the coordinate is a nominal value, it mutates to another one, among all the possible values of the attribute or feature, with a probability inversely proportional to the HVDM (Wilson and Martinez, 1997) distance from the original value. The number of coordinates to be mutated is randomly obtained and is a number below 25% of the total number of features.
- Operator *Biased Mutation*: modifies the width and all coordinates of the centre using local information of the RBF environment. A clustering-based technique for training centres has been used. In this way the RBF centre, $\vec{c}_i$, is modified as follows:

$$c'_{ij} = c_{ij} \pm h \quad \forall j = 1,\ldots,n. \tag{10}$$

The increase or decrease of the old centre is decided by means of a random number h ($h \leqslant 0.5 \cdot d_i$). The centre is varied in order to approximate it to the average of the patterns belonging to the RBF class and inside its RBF width. The objective of the width training is that most of the patterns belonging to the RBF class will be inside the RBF width. In this way the RBF width is modified as follows:

$$\begin{cases} d' = d - h & \text{if} \quad u \leqslant D \\ d' = d + h & \text{otherwise} \end{cases} \quad D = \frac{npnci}{npci} \quad A = \frac{npci2}{npnci2}, \tag{11}$$

where h is a random number ($h \leqslant 0.5 \cdot d_i$); u is a random number ($u \leqslant D + A$); $npnci$ is the number of patterns not belonging to the RBF class inside the RBF width; $npci$ is the number of patterns belonging to the RBF class inside the RBF width; $npci2$, is the number of patterns belonging to the RBF class inside twice RBF width and $npnci2$, is the number of patterns not belonging to the RBF class inside twice RBF width.
- Operator *Null*: in this case all the parameters of the RBF are maintained.

With these mutation operators CO^{2}RBFN promotes an appropriate balance between exploitation and exploration, which is a desirable feature in every evolutionary algorithm. Biased mutations use local information from the RBF environment in order to achieve an optimal adaptation. On the other hand, random mutations carry out alterations that lead to the exploration of the environment and thus avoid local optimums.

The operators are applied to the whole population of RBFs. The probability for choosing an operator is determined by means of a Mandani-type FRBS (Mandani and Assilian, 1975), which represents expert knowledge about the operator application in order to obtain a simple and accurate RBFN.

The inputs of this system are parameters a_i, e_i and o_i used for defining the credit assignment of the RBF ϕ_i. These inputs are considered as linguistic variables va_i, ve_i and vo_i. The outputs, p_{remove}, p_{rm}, p_{bm} and p_{null}, represent the probability of applying *Remove, Random Mutation, Biased Mutation* and *Null* operators, respectively. The number of linguistic labels has been empirically determined and the fuzzy sets have been defined according to their meaning. Fig. 6 shows the membership functions for the input and output variables respectively. Table 2 shows the rule base used to relate the described antecedents and consequents. In the table each row represents one rule. For example, the interpretation of the first rule is: If the contribution of an RBF is Low Then the probability of applying the operator Remove is Medium–High, the probability of

Table 2
Fuzzy rule base representing expert knowledge in the design of RBFNs.

	Antecedents			Consequents			
	v_a	v_e	v_o	p_{remove}	p_{rm}	p_{bm}	p_{null}
R1	L			M–H	M–H	L	L
R2	M			M–L	M–H	M–L	M–L
R3	H			L	M–H	M–H	M–H
R4		L		L	M–H	M–H	M–H
R5		M		M–L	M–H	M–L	M–L
R6		H		M–H	M–H	L	L
R7			L	L	M–H	M–H	M–H
R8			M	M–L	M–H	M–L	M–L
R9			H	M–H	M–H	L	L

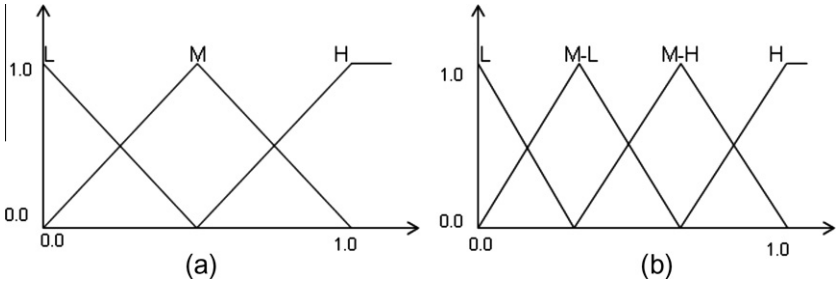

Fig. 6. (a) input variables membership functions for the FRBS. (b) Output variables membership function.

applying the operator Random Mutation is Medium–High, the probability of applying the operator Biased Mutation is Low and the probability of applying the operator null is Low.

The rule base represents expert knowledge, as mentioned, in the design of RBFNs. It was developed taking into account the fact that an RBF is worse if its contribution (a_i) is low, its error (e_i) is high and its overlapping (o_i) is also high. On the other hand, an RBF is better when its contribution is high, its error is low and its overlapping is also low. A worse RBF indicates that this neuron has problems performing a good role in its environment and therefore important changes such as random mutations or even removing the RBF must be promoted. In these cases the probability of applying the biased mutation operator and the null operator is low. However, a better neuron implies that the RBF is working well in its environment. In these situations exploitation is promoted by increasing the probability of applying the biased mutation operator. The probability of maintaining the neuron with the same parameters, applying the null operator, is also augmented. In these cases the probability of removing the RBF will be low. The probability of applying random mutation is usually high in order to promote a parsimonious evolution. It can be highlighted that this rule base represents general knowledge related to the design of RBFNs. This generic rule base has been successfully applied to classification problems (Pérez-Godoy et al., 2010).

4.5. Introduction of new RBFs

In this step of the algorithm, the eliminated RBFs are substituted by new RBFs. The new RBF is located in the centre of the area with maximum error or in a randomly chosen pattern with a probability of 0.5, respectively.

In the first instance, the areas are defined as a set of neighbouring patterns with a width equal to the average of the width of the RBF. The width of the new RBF will be set to the average of the RBFs in the population plus half of the minimum distance to the nearest RBF. Its weights are set to zero.

If it is chosen randomly, the RBF is located in the first pattern found outside of any RBF width. The width of the new RBF is set to the average of the RBFs in the population, and its weights are set to zero.

4.6. Replacement strategy

After applying the mutation operators, new RBFs appear. The algorithm uses the replacement scheme to determine which new RBFs will be included in the new population. To do so, the role of the mutated RBF in the network is compared with the original one to determine the RBF with the best behaviour in order to include it in the population.

Table 3
Description for imbalanced data sets.

Data sets	#Ex.	#Atts.	Class (minority, majority)	% Class (minority, majority)	IR
Glass1	214	9	(build-win-non-float-proc, remainder)	(35.51, 64.49)	1.82
Ecoli0vs1	220	7	(im, cp)	(35.00, 65.00)	1.86
Wisconsin	683	9	(malignant, benign)	(35.00, 65.00)	1.86
Pima	768	8	(tested-positive, tested-negative)	(34.84, 66.16)	1.90
Iris0	150	4	(iris-setosa, remainder)	(33.33, 66.67)	2.00
Glass0	214	9	(build-win-float-proc, remainder)	(32.71, 67.29)	2.06
Yeast1	1484	8	(nuc, remainder)	(28.91, 71.09)	2.46
Vehicle1	846	18	(saab, remainder)	(28.37, 71.63)	2.52
Vehicle2	846	18	(bus, remainder)	(28.37, 71.63)	2.52
Vehicle3	846	18	(opel, remainder)	(28.37, 71.63)	2.52
Haberman	306	3	(die, survive)	(27.42, 73.58)	2.68
Glass0123vs456	214	9	(non-window glass, remainder)	(23.83, 76.17)	3.19
Vehicle0	846	18	(van, remainder)	(23.64, 76.36)	3.23
Ecoli1	336	7	(im, remainder)	(22.92, 77.08)	3.36
New-thyroid2	215	5	(hypo, remainder)	(16.89, 83.11)	4.92
New-thyroid1	215	5	(hyper, remainder)	(16.28, 83.72)	5.14
Ecoli2	336	7	(pp, remainder)	(15.48, 84.52)	5.46
Segment0	2308	19	(brickface, remainder)	(14.26, 85.74)	6.01
Glass6	214	9	(headlamps, remainder)	(13.55, 86.45)	6.38
Yeast3	1484	8	(me3, remainder)	(10.98, 89.02)	8.11
Ecoli3	336	7	(imU, remainder)	(10.88, 89.12)	8.19
Page-blocks0	5472	10	(remainder, text)	(10.23, 89.77)	8.77
Yeast2vs4	514	8	(cyt, me2)	(9.92, 90.08)	9.08
Yeast05679vs4	528	8	(me2, mit, me3, exc, vac, erl)	(9.66, 90.34)	9.35
Vowel0	988	13	(hid, remainder)	(9.01, 90.99)	10.10
Glass016vs2	192	9	(ve-win-float-proc, build-win-float-proc, build-win-non-float-proc, headlamps)	(8.89, 91.11)	10.29
Glass2	214	9	(ve-win-float-proc, remainder)	(8.78, 91.22)	10.39
Ecoli4	336	7	(om, remainder)	(6.74, 93.26)	13.84
Yeast1vs7	459	8	(vac, nuc)	(6.72, 93.28)	13.87
Shuttle0vs4	1829	9	(rad-flow, bypass)	(6.72, 93.28)	13.87
Glass4	214	9	(containers, remainder)	(6.07, 93.93)	15.47
Page-blocks13vs2	472	10	(graphic, horiz.line, picture)	(5.93, 94.07)	15.85
Abalone9vs18	731	8	(18, 9)	(5.65, 94.25)	16.68
Glass016vs5	184	9	(tableware, build-win-float-proc, build-win-non-float-proc, headlamps)	(4.89, 95.11)	19.44
Shuttle2vs4	129	9	(fpv Open, Bypass)	(4.65, 95.35)	20.5
Yeast1458vs7	693	8	(vac, nuc, me2, me3, pox)	(4.33, 95.67)	22.10
Glass5	214	9	(tableware, remainder)	(4.20, 95.80)	22.81
Yeast2vs8	482	8	(pox, cyt)	(4.15, 95.85)	23.10
Yeast4	1484	8	(me2, remainder)	(3.43, 96.57)	28.41
Yeast1289vs7	947	8	(vac, nuc, cyt, pox, erl)	(3.17, 96.83)	30.56
Yeast5	1484	8	(me1, remainder)	(2.96, 97.04)	32.78
Ecoli0137vs26	281	7	(pp, imL, cp, im, imU, imS)	(2.49, 97.51)	39.15
Yeast6	1484	8	(exc, remainder)	(2.49, 97.51)	39.15
Abalone19	4174	8	(19, remainder)	(0.77, 99.23)	128.87

5. Experimental framework

In this section we first describe the collection of imbalanced data sets selected for our study (Section 5.1). Then, we show the algorithms selected for comparison in the experimental study and the corresponding parameters (Section 5.2). Finally, we present the statistical tests used in all our analysis (Section 5.3).

5.1. Data sets

In this study CO^{2}RBFN is applied to forty-four binary data sets from the UCI repository (Asuncion and Newman, 2007) with different imbalance ratio (IR) (Orriols-Puig and Bernadó-Mansilla, 2009). Table 3 summarises the data selected in this study and shows, for each data set, the number of examples (#Ex.), number of attributes (#Atts.), class name of each class (minority and majority), class attribute distribution and IR. This table is ordered by the IR, from low to highly imbalanced data sets.

In order to estimate precision we use a fivefold cross validation approach, that is five partitions for training and test sets, 80% for training and 20% for testing, where the five test partitions form the whole set. For each data set we consider the average results of the five partitions.

5.2. Algorithms for comparison and parameters

Regarding the algorithms for comparison, we have selected alternative paradigms in the RBFN design field, other neural network models such as multilayer perceptron and learning vector quantization network, and rule induction algorithms. Specifically, we consider five different neural network approaches:

- *LVQ:* builds a learning vector quantization network composed of a set of neurons. The set of neurons represents the most indicative prototypes for each class after training, so the class of each instance will be predicted as the class of the nearest neuron, following a KNN model (Bezdek and Kuncheva, 2001).
- *MLP–Back:* algorithm for multilayer perceptron networks design which uses the backpropagation algorithm for learning (Rojas and Feldman, 1996).
- *MLP–Grad:* algorithm for multilayer perceptron networks design which uses the gradient descent algorithm for learning (Moller, 1990; Widrow and Lehr, 1990).
- *RBFN–Decr:* algorithm for RBFNs design based on a decremental scheme (Broomhead and Lowe, 1988).
- *RBFN–Incr:* algorithm for RBFNs design based on an incremental scheme (Plat, 1991).

Furthermore, we have selected the C4.5 algorithm (Quilan, 1993) as a well-known classifier that has been widely used for imbalanced data (Orriols-Puig and Bernadó-Mansilla, 2009; Su et al., 2006; Su and Hsiao, 2007), and a HFRBCS (Fernández et al., 2009), which is a fuzzy classifier that has shown to be very competitive within this framework.

HFRBCS has been developed by the authors and the remaining methods were run using KEEL software (Alcalá-Fdez et al., 2009), following the recommended parameter values given in the KEEL platform to configure the methods, which also correspond to the settings used in the bibliography of these methods. Table 4 summarises the parameters for the different approaches used in the experimental study.

Regarding the use of the SMOTE pre-processing method (Chawla et al., 2002), we consider only the 1-nearest neighbour (using the euclidean distance) to generate the synthetic samples, and we balance both classes to the 50% distribution.

Table 4
Parameter specification for the algorithms employed in the experimentation.

Algorithm	Parameter	Value
C4.5	Pruned	True
	Confidence	0.25
	InstancesPerLeaf	2
CO^{2}RBFN	Generations of the main loop	200
	Number of RBFs	5
HFRBCS	*Fuzzy configuration*	
	Conjunction operator	Product *T*-norm
	Rule weight heuristic	Penalised certainty factor
	Inference mechanism	Winning rule
	Hierarchical Generation Process	
	Alpha	0.2
	Number of evaluations	10,000
	Population size	61
LVQ	Iterations	100
	Neurons	20
	Alpha	0.3
	Nu	0.8
MLP–Back	Hidden-layer	2
	Hidden-nodes	15
	Transfer	Htan
	Eta	0.15
	Alpha	0.10
	Lambda	0.0
MLP–Grad	Hidden-nodes	10
RBFN–Decr	Percent	0.1
	Initial neurons	20
	Alpha	0.3
RBFN–Incr	Epsilon	0.1
	Alpha	0.3
	Delta	0.5

5.3. Statistical tests for evaluation

In this paper, we use the hypothesis testing techniques to provide statistical support to the analysis of the results (García et al., 2009; Sheskin, 2006). Specifically, we will use non-parametric tests due to the fact that the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, causing the statistical analysis to lose credibility with these parametric tests (Demšar, 2006).

We will use the Wilcoxon signed-rank test (Wilcoxon, 1945) as a non-parametric statistical procedure for performing pairwise comparisons between two algorithms. For multiple comparisons we use the Iman–Davenport test (Sheskin, 2006) to detect statistical differences among a group of results, and the Holm post hoc test (Holm, 1979) in order to find which algorithms are distinctive among a $1 \times n$ comparison.

The post hoc procedure allows us to know whether a hypothesis of comparison of means could be rejected at a specified level of significance α. However, it is very interesting to compute the *p*-value associated to each comparison, which represents the lowest level of significance of a hypothesis that results in a rejection. In this manner, we can know whether two algorithms are significantly different and how different they are.

Furthermore, we consider the average ranking of the algorithms in order to show graphically how good a method is with respect to its partners. This ranking is obtained by assigning a position to each algorithm depending on its performance for each data set. The algorithm which achieves the best accuracy on a specific data set will have the first ranking (value 1); then, the algorithm with the second best accuracy is assigned rank 2, and so forth. This task is carried out for all data sets and finally an average ranking is computed as the mean value of all rankings.

Table 5
Experimentation results without pre-processing.

Base datos	C4.5	CO²RBFN	HFRBCS	LVQ	MLP–Back	MLP–Grad	RBFN–Decr	RBFN–Incr
Glass1	71.45 ± 5.40	69.30 ± 6.16	49.65 ± 5.35	62.27 ± 11.19	56.32 ± 10.53	71.28 ± 4.47	67.99 ± 10.07	73.23 ± 8.10
Ecoli0vs1	98.31 ± 2.39	97.03 ± 2.80	94.43 ± 5.49	93.68 ± 3.87	97.59 ± 1.99	0.00 ± 0.00	94.71 ± 4.23	92.93 ± 8.40
Wisconsin	94.50 ± 3.20	97.29 ± 0.77	87.99 ± 2.85	92.65 ± 5.78	96.23 ± 1.15	93.89 ± 2.23	95.31 ± 1.88	95.55 ± 1.89
Pima	68.72 ± 2.95	71.73 ± 4.30	63.88 ± 5.06	56.95 ± 10.07	71.10 ± 4.80	67.72 ± 4.14	65.00 ± 6.63	62.03 ± 5.48
Iris0	98.97 ± 2.29	99.79 ± 1.01	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.70 ± 0.82	99.80 ± 0.69
Glass0	81.36 ± 3.05	75.69 ± 7.12	62.85 ± 4.21	69.58 ± 7.76	64.68 ± 8.60	80.08 ± 6.44	68.01 ± 15.37	70.76 ± 12.06
Yeast1	62.83 ± 5.74	70.77 ± 3.58	48.28 ± 2.11	53.62 ± 8.61	65.86 ± 4.92	63.36 ± 2.91	39.71 ± 18.55	46.94 ± 15.37
Vehicle1	95.57 ± 1.26	65.57 ± 2.98	86.87 ± 3.38	52.18 ± 13.27	60.85 ± 13.43	77.10 ± 4.55	52.45 ± 18.91	59.36 ± 6.46
Vehicle2	64.70 ± 2.56	83.37 ± 3.83	56.82 ± 6.03	69.53 ± 8.85	63.86 ± 11.07	97.65 ± 1.71	72.50 ± 8.73	91.23 ± 3.46
Vehicle3	61.37 ± 6.22	66.97 ± 3.46	56.31 ± 5.56	52.75 ± 8.07	64.15 ± 4.84	74.68 ± 5.70	56.27 ± 10.93	54.42 ± 6.37
Haberman	35.63 ± 32.87	61.21 ± 7.26	23.47 ± 14.20	42.08 ± 12.97	61.10 ± 8.59	45.74 ± 6.85	47.62 ± 12.55	47.99 ± 9.51
Glass0123vs456	91.31 ± 7.98	92.27 ± 3.27	88.51 ± 6.45	85.88 ± 8.21	91.55 ± 3.44	86.15 ± 4.52	87.74 ± 7.08	93.63 ± 4.39
Vehicle0	92.90 ± 3.21	89.12 ± 4.55	84.62 ± 3.17	67.90 ± 12.22	68.54 ± 11.90	95.70 ± 2.43	79.21 ± 14.20	94.09 ± 2.70
Ecoli1	85.38 ± 3.75	88.65 ± 4.42	86.09 ± 5.23	77.00 ± 10.51	85.05 ± 3.83	66.92 ± 34.41	79.20 ± 13.93	82.17 ± 8.92
Newthyroid2	93.53 ± 2.89	98.40 ± 3.72	77.72 ± 17.51	78.73 ± 25.19	84.16 ± 18.26	95.93 ± 4.32	90.89 ± 6.38	98.86 ± 2.32
Newthyroid1	90.84 ± 6.52	98.02 ± 3.05	76.42 ± 14.12	84.86 ± 12.48	82.81 ± 19.05	96.49 ± 3.78	92.98 ± 6.21	97.99 ± 3.73
Coli2	85.14 ± 11.24	92.02 ± 3.40	91.04 ± 2.55	85.89 ± 11.71	81.11 ± 8.22	68.50 ± 34.87	77.11 ± 15.43	78.67 ± 14.51
Segment0	98.24 ± 1.18	96.05 ± 2.20	97.39 ± 1.49	82.61 ± 9.12	1.71 ± 8.36	0.00 ± 0.00	59.55 ± 23.33	98.22 ± 1.46
Glass6	79.42 ± 7.05	87.07 ± 7.38	85.46 ± 9.15	85.75 ± 8.04	90.40 ± 6.42	88.60 ± 11.01	88.12 ± 8.95	91.23 ± 6.60
Yeast3	85.10 ± 6.81	89.51 ± 2.58	48.24 ± 8.05	67.78 ± 18.86	74.57 ± 5.51	81.82 ± 6.51	32.65 ± 24.84	67.40 ± 14.61
Ecoli3	67.38 ± 13.36	87.02 ± 7.65	69.85 ± 13.47	74.24 ± 19.72	59.14 ± 33.33	54.32 ± 29.03	66.64 ± 23.66	68.43 ± 29.52
Pageblocks0	91.95 ± 2.38	86.07 ± 2.40	66.70 ± 5.69	55.39 ± 15.72	73.72 ± 8.05	85.01 ± 1.85	65.00 ± 15.38	86.58 ± 2.43
Yeast2vs4	81.69 ± 4.03	86.87 ± 4.90	72.83 ± 14.71	73.54 ± 11.47	72.81 ± 6.48	65.44 ± 34.06	67.13 ± 25.15	70.71 ± 19.93
Yeast05679vs4	60.53 ± 17.58	77.06 ± 8.20	27.69 ± 16.45	64.12 ± 12.75	66.31 ± 10.08	58.25 ± 11.16	52.06 ± 18.50	40.14 ± 20.82
Vowel0	96.83 ± 6.63	87.03 ± 5.86	98.86 ± 2.56	51.52 ± 14.27	68.74 ± 6.33	98.97 ± 1.79	83.81 ± 12.29	99.43 ± 1.39
Glass016vs2	42.16 ± 23.87	47.27 ± 19.50	0.00 ± 0.00	32.94 ± 31.01	23.99 ± 27.91	46.90 ± 29.85	21.98 ± 26.97	29.92 ± 31.83
Glass2	60.08 ± 35.63	57.23 ± 15.11	0.00 ± 0.00	27.58 ± 26.77	18.89 ± 21.63	28.60 ± 30.78	23.86 ± 27.24	26.67 ± 28.65
Ecoli4	82.33 ± 11.77	90.96 ± 6.23	75.97 ± 16.02	78.49 ± 15.70	62.30 ± 32.00	69.09 ± 34.55	82.12 ± 15.88	80.65 ± 15.93
Shuttlec0vsc4	99.97 ± 0.07	69.69 ± 12.50	99.12 ± 1.15	96.13 ± 10.16	82.64 ± 24.84	99.60 ± 0.81	98.53 ± 5.42	99.75 ± 0.55
Yeast1vs7	45.10 ± 27.42	99.67 ± 0.80	16.33 ± 22.36	36.75 ± 28.45	53.80 ± 16.04	38.51 ± 23.65	35.90 ± 23.42	6.53 ± 14.97
Glass4	55.37 ± 51.05	81.84 ± 14.07	52.60 ± 31.13	65.47 ± 24.22	79.66 ± 18.82	71.70 ± 25.45	72.06 ± 34.50	94.89 ± 8.68
Pageblocks13vs4	99.77 ± 0.51	90.15 ± 7.70	57.25 ± 35.03	74.97 ± 11.49	88.21 ± 9.29	98.66 ± 4.04	45.81 ± 27.89	83.41 ± 11.33
Abalone918	44.82 ± 9.83	75.70 ± 9.52	7.07 ± 15.81	27.54 ± 22.45	55.56 ± 18.66	62.98 ± 11.28	32.60 ± 15.58	20.62 ± 16.85
Glass016vs5	79.42 ± 44.41	62.40 ± 40.00	47.29 ± 44.61	35.92 ± 41.56	87.52 ± 4.55	79.55 ± 35.97	71.78 ± 29.02	80.06 ± 21.55
Shuttlec2vsc4	94.14 ± 13.10	93.59 ± 11.70	92.48 ± 12.43	70.14 ± 40.99	80.28 ± 18.41	94.66 ± 10.51	55.25 ± 49.00	81.59 ± 36.05
Yeast1458vs7	0.00 ± 0.00	55.02 ± 14.70	0.00 ± 0.00	21.76 ± 21.34	45.10 ± 17.16	8.75 ± 17.76	3.89 ± 13.45	0.00 ± 0.00
Glass5	88.04 ± 15.83	57.30 ± 43.95	0.00 ± 0.00	21.89 ± 35.73	85.28 ± 5.84	88.51 ± 21.88	66.02 ± 42.34	67.24 ± 39.76
Yeast2vs8	10.00 ± 22.36	71.88 ± 14.13	69.58 ± 12.85	60.66 ± 29.22	64.18 ± 15.59	70.72 ± 19.06	71.76 ± 13.96	72.79 ± 13.42
Yeast4	41.97 ± 27.02	77.33 ± 10.86	12.65 ± 17.32	45.02 ± 20.08	60.29 ± 14.76	44.57 ± 8.61	30.71 ± 26.18	12.94 ± 17.64
Yeast1289vs7	41.97 ± 26.64	55.19 ± 22.50	8.16 ± 18.26	26.13 ± 22.28	38.00 ± 23.10	42.08 ± 23.47	18.45 ± 23.43	0.00 ± 0.00
Yeast5	87.51 ± 6.31	94.12 ± 4.40	48.29 ± 12.81	76.93 ± 23.64	59.79 ± 26.63	63.98 ± 14.45	35.99 ± 29.21	44.80 ± 23.77
Ecoli0137vs26	56.60 ± 36.24	70.50 ± 29.50	35.77 ± 23.31	52.16 ± 46.90	61.92 ± 35.45	58.83 ± 48.37	68.70 ± 40.14	69.50 ± 40.62
Yeast6	54.01 ± 50.80	83.27 ± 10.45	73.65 ± 43.09	64.85 ± 28.28	60.33 ± 16.06	53.23 ± 20.10	15.33 ± 25.68	28.34 ± 30.40
Abalone19	0.00 ± 0.00	50.12 ± 21.81	0.00 ± 0.00	11.42 ± 20.60	49.88 ± 13.87	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Mean	70.84 ± 12.85	**79.48 ± 9.46**	56.78 ± 10.93	61.53 ± 17.76	67.27 ± 13.18	66.69 ± 13.62	59.82 ± 18.03	65.03 ± 12.80

Algorithm	Ranking
C4.5	3.727
CO²RBFN	**2.364**
HFRBCS	5.943
LVQ	5.534
MLP-Back	4.352
MLP-Grad	4.034
RBFN-Decr	5.705
RBFN-Incr	4.341

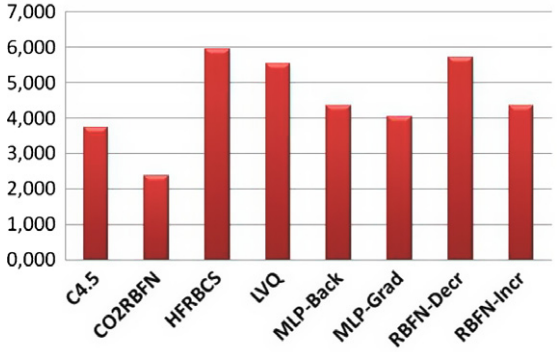

Fig. 7. Ranking according to the GM performance for all algorithms without pre-processing. The lower the value, the better.

These tests are suggested in the studies presented in (Demšar, 2006; García and Herrera, 2008; García et al., 2009; García et al., 2010), where their use in the field of machine learning is highly recommended. For a wider description of the use of these tests, please refer to Appendix A. Furthermore, any interested reader can find additional information on the website: http://sci2s.ugr.es/sicidm/, together with the software for applying the statistical tests.

6. Analysis of the results

In this section we will carry out a complete experimental analysis in order to show two important issues:

1. First, the performance of the algorithms when they are applied over the original data sets (Section 6.1).

Table 6
Holm test table without pre-processing in all imbalanced data sets. CO^2RBFN is the control method.

i	Algorithm	z	p	$\alpha\backslash i$	Hypothesis ($\alpha = 0.05$)
7	HFRBCS	6.854	7.166E−12	0.0071	Rejected
6	RBFN–Decr	6.397	1.581E−10	0.0083	Rejected
5	LVQ	6.071	1.271E−9	0.01	Rejected
4	RBFN–Back	3.808	1.401	0.0125	Rejected
3	MLP–Incr	3.786	1.530	0.0167	Rejected
2	MLP–Grad	3.199	0.001	0.025	Rejected
1	C4.5	2.611	0.009	0.05	Rejected

2. Then, the synergy between the SMOTE pre-processing and the algorithms used in this paper is studied (Section 6.2).

6.1. Performance analysis without pre-processing (original data sets)

Following, we analyse the performance of the methods considering all the original, without pre-processing, data sets. The complete table of results for all the algorithms used in this study is shown in Table 5, where the reader can observe the full test results, with their associated standard deviation, in order to compare the performance of each approach, the GM metric is used. We must emphasise the good results achieved by CO^2RBFN, as it obtains the highest value among all algorithms.

In order to analyse these results, Fig. 7 shows the average ranking computed for all approaches according to the GM metric, where we can observe that CO^2RBFN has obtained the lowest value in the ranking and therefore it is the best algorithm.

Next, we check for significant differences among the results of the algorithms used in the experimental study by means of an Iman and Davenport test. This obtains a p-value near to zero, which implies that there are significant differences among the results and therefore we should apply a post hoc test to detect which methods are outperformed by CO^2RBFN, since it is the best ranked method.

Specifically, we apply a Holm test to compare the best ranking method (CO^2RBFN) with the remaining methods. The result is shown in Table 6, in which the algorithms are ordered with respect to the z value obtained. The value p_i obtained is compared with the value $\alpha\backslash i$ in the same row of the table, in every case the value p_i, and is lower than the $\alpha\backslash i$ corresponding, which implies there are significant differences between the control algorithm, CO^2RBFN, and the other algorithms.

Table 7
Experimentation results with SMOTE.

Base datos	C4.5	CO^2RBFN	HFRBCS	LVQ	MLP–Back	MLP–Grad	RBFN–Decr	RBFN–Incr
Glass1	75.11 ± 3.74	69.86 ± 6.44	73.66 ± 4.66	65.33 ± 7.63	57.53 ± 7.14	71.93 ± 5.98	67.95 ± 14.41	73.78 ± 8.91
Ecoli0vs1	97.95 ± 2.20	96.18 ± 2.96	93.63 ± 6.45	93.05 ± 3.63	97.22 ± 1.70	0.00 ± 0.00	94.69 ± 4.71	95.13 ± 5.26
Wisconsin	95.44 ± 2.01	97.26 ± 0.85	88.24 ± 1.63	94.02 ± 5.18	90.52 ± 17.72	95.31 ± 1.57	94.41 ± 4.95	94.16 ± 5.07
Pima	71.26 ± 4.05	72.56 ± 3.71	68.72 ± 5.26	59.63 ± 5.77	69.89 ± 6.63	69.62 ± 4.85	63.97 ± 7.71	61.95 ± 4.45
Iris0	98.97 ± 2.29	99.90 ± 0.50	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.80 ± 0.69	99.69 ± 1.10
Glass0	78.14 ± 2.21	75.64 ± 6.99	76.57 ± 8.05	67.79 ± 5.03	68.62 ± 6.21	79.53 ± 5.75	71.89 ± 9.37	76.36 ± 6.62
Yeast1	70.86 ± 2.95	70.08 ± 3.47	71.71 ± 2.39	57.77 ± 6.70	63.16 ± 13.54	72.34 ± 3.11	34.15 ± 20.61	62.45 ± 9.80
Vehicle1	94.85 ± 1.68	69.08 ± 4.37	90.61 ± 2.17	59.21 ± 6.01	61.19 ± 5.93	81.94 ± 3.31	61.78 ± 9.60	60.70 ± 5.03
Vehicle2	69.28 ± 3.41	87.24 ± 3.98	71.76 ± 2.64	72.34 ± 5.75	70.18 ± 8.18	97.50 ± 1.58	73.19 ± 10.18	91.55 ± 2.98
Vehicle3	74.34 ± 1.08	69.55 ± 3.98	66.80 ± 3.34	60.27 ± 5.96	64.82 ± 3.06	79.27 ± 3.49	61.28 ± 6.82	58.37 ± 6.79
Haberman	61.32 ± 3.85	60.21 ± 6.25	57.08 ± 4.09	50.92 ± 8.28	56.24 ± 10.64	57.47 ± 7.29	58.04 ± 8.51	53.38 ± 5.21
Glass0123vs456	90.13 ± 3.17	93.78 ± 3.28	88.37 ± 3.97	88.46 ± 5.58	84.03 ± 7.56	87.67 ± 5.65	88.94 ± 9.22	92.17 ± 4.33
Vehicle0	91.10 ± 2.70	92.15 ± 2.48	88.92 ± 1.96	77.65 ± 5.05	81.40 ± 5.02	95.07 ± 2.70	73.22 ± 19.41	90.21 ± 5.11
Ecoli1	76.10 ± 9.58	87.84 ± 4.10	84.18 ± 12.69	85.03 ± 5.72	86.85 ± 3.88	69.58 ± 35.02	81.50 ± 18.02	87.57 ± 6.31
Newthyroid2	96.51 ± 4.87	98.46 ± 2.22	99.72 ± 0.63	91.34 ± 7.51	98.48 ± 1.12	98.82 ± 3.00	93.52 ± 5.17	99.04 ± 1.37
Newthyroid1	97.98 ± 3.79	97.54 ± 4.03	98.58 ± 2.48	94.95 ± 4.83	97.35 ± 2.63	99.44 ± 0.69	88.98 ± 7.35	98.63 ± 2.43
Ecoli2	91.60 ± 4.86	93.14 ± 4.50	87.62 ± 8.24	83.62 ± 7.41	87.54 ± 6.36	72.93 ± 36.58	72.68 ± 21.91	81.85 ± 15.80
Segment0	99.26 ± 0.61	97.97 ± 0.81	97.51 ± 1.11	82.45 ± 6.08	1.99 ± 9.77	0.00 ± 0.00	62.58 ± 14.89	97.96 ± 1.15
Glass6	83.00 ± 9.05	85.93 ± 8.39	86.95 ± 10.84	85.86 ± 7.19	88.05 ± 7.17	85.19 ± 9.20	88.16 ± 8.22	85.52 ± 9.05
Yeast3	88.50 ± 3.66	91.11 ± 2.34	90.41 ± 2.34	82.92 ± 4.88	82.96 ± 17.18	91.65 ± 2.47	56.92 ± 29.97	87.77 ± 4.56
Ecoli3	88.77 ± 7.65	85.72 ± 7.90	90.81 ± 4.43	82.27 ± 6.06	86.65 ± 6.09	68.36 ± 34.94	78.96 ± 20.01	86.26 ± 9.65
Pageblocks0	94.84 ± 1.52	88.60 ± 2.01	91.40 ± 0.67	76.39 ± 4.95	78.65 ± 7.02	93.72 ± 1.05	68.33 ± 16.66	57.03 ± 13.20
Yeast2vs4	85.09 ± 10.15	87.29 ± 3.60	89.32 ± 4.18	82.79 ± 5.09	85.61 ± 6.06	67.96 ± 34.57	69.93 ± 23.54	81.72 ± 10.33
Yeast05679vs4	74.88 ± 10.88	78.22 ± 5.10	73.18 ± 7.47	69.77 ± 9.12	76.90 ± 5.61	75.32 ± 6.46	53.02 ± 27.34	72.19 ± 12.11
Vowel0	94.74 ± 5.22	93.77 ± 3.52	98.82 ± 1.62	78.12 ± 6.18	84.55 ± 6.61	99.19 ± 1.73	93.14 ± 10.87	99.36 ± 1.63
Glass016vs2	48.91 ± 29.44	56.44 ± 20.80	58.37 ± 20.04	58.93 ± 12.85	45.07 ± 26.29	62.97 ± 18.08	66.17 ± 18.50	64.96 ± 14.63
Glass2	33.86 ± 32.29	58.15 ± 25.25	54.84 ± 20.57	58.76 ± 13.33	63.98 ± 10.68	67.51 ± 17.36	62.63 ± 17.59	66.42 ± 11.93
Ecoli4	81.28 ± 11.67	89.65 ± 6.17	93.02 ± 8.17	92.37 ± 5.29	91.35 ± 6.65	68.62 ± 34.32	88.05 ± 18.95	93.72 ± 5.94
Shuttlec0vsc4	99.97 ± 0.07	99.58 ± 0.80	99.12 ± 1.15	99.60 ± 0.81	99.83 ± 0.55	99.75 ± 0.66	98.27 ± 5.48	99.91 ± 0.12
Yeast1vs7	67.73 ± 2.28	99.49 ± 0.90	70.74 ± 12.40	59.38 ± 15.49	69.75 ± 8.32	62.94 ± 12.31	42.46 ± 25.31	64.85 ± 11.41
Glass4	83.71 ± 10.78	85.45 ± 12.62	70.39 ± 40.49	80.68 ± 20.55	87.93 ± 10.08	80.94 ± 26.70	76.64 ± 25.73	93.62 ± 8.31
Pageblock13vs4	99.55 ± 0.47	96.65 ± 3.60	98.64 ± 0.65	93.29 ± 4.61	79.88 ± 12.93	98.50 ± 2.76	88.33 ± 18.54	86.55 ± 13.99
Abalone918	15.58 ± 21.36	77.41 ± 10.60	70.19 ± 8.56	52.11 ± 11.77	75.41 ± 16.73	75.42 ± 11.58	53.82 ± 9.78	70.30 ± 8.91
Glass016vs5	72.08 ± 42.33	84.71 ± 31.80	77.96 ± 43.61	85.09 ± 12.39	91.21 ± 6.84	85.16 ± 27.25	85.27 ± 12.36	83.58 ± 22.37
Shuttlec2vsc4	99.15 ± 1.90	99.51 ± 1.00	97.49 ± 2.71	97.92 ± 4.48	97.93 ± 5.06	99.26 ± 1.28	77.14 ± 38.80	75.25 ± 38.66
Yeast1458vs7	41.19 ± 6.06	60.80 ± 11.20	62.49 ± 6.26	52.49 ± 12.66	60.79 ± 7.76	55.12 ± 20.49	34.52 ± 22.68	52.03 ± 17.95
Glass5	86.70 ± 15.44	74.91 ± 38.45	68.73 ± 39.56	88.73 ± 7.60	80.49 ± 30.59	77.83 ± 31.32	69.87 ± 36.50	69.82 ± 37.32
Yeast2vs8	78.23 ± 13.05	77.31 ± 12.23	72.47 ± 15.10	69.26 ± 10.47	71.19 ± 11.67	70.06 ± 21.25	53.58 ± 25.15	74.19 ± 12.05
Yeast4	65.00 ± 8.94	78.95 ± 4.23	82.64 ± 2.29	76.50 ± 6.58	80.92 ± 4.56	77.96 ± 8.51	59.46 ± 21.78	78.62 ± 7.27
Yeast1289vs7	64.13 ± 9.00	70.14 ± 7.50	69.37 ± 4.37	55.15 ± 8.35	69.44 ± 5.48	60.87 ± 9.10	23.28 ± 24.96	63.19 ± 10.36
Yeast5	92.04 ± 4.99	94.69 ± 3.60	94.20 ± 2.59	94.86 ± 2.07	91.67 ± 6.02	93.17 ± 5.09	40.41 ± 43.18	94.54 ± 3.96
Ecoli0137vs26	80.38 ± 15.47	70.09 ± 36.30	84.92 ± 12.88	66.41 ± 34.27	54.87 ± 45.05	57.69 ± 47.13	62.74 ± 38.02	61.16 ± 43.29
Yeast6	71.21 ± 41.31	86.57 ± 8.40	71.48 ± 41.80	76.41 ± 10.18	85.78 ± 5.61	84.59 ± 7.71	34.09 ± 36.30	83.47 ± 9.04
Abalone19	53.19 ± 8.25	70.18 ± 11.77	67.56 ± 14.01	52.86 ± 15.69	60.46 ± 14.83	51.87 ± 11.45	54.47 ± 14.09	63.32 ± 12.10
Mean	78.95 ± 8.69	**83.40 ± 7.84**	81.57 ± 9.10	76.20 ± 8.07	76.78 ± 9.29	75.91 ± 11.94	68.69 ± 17.81	79.19 ± 10.18

Table 8
Wilcoxon test to compare the use of SMOTE with the performance without pre-processing. R^+ corresponds to the SMOTE pre-processing and R^- to the original data sets.

Comparison	R^+	R^-	p-Value	Hypothesis ($\alpha = 0.05$)
CO^{2}RBFN + SMOTE vs. CO^{2}RBFN	856.0	134.0	0.000	Rejected for CO^{2}RBFN + SMOTE
LVQ + SMOTE vs. LVQ	969.5	20.5	0.000	Rejected for LVQ + SMOTE
MLP–Back + SMOTE vs. MLP–Back	853.5	136.5	0.000	Rejected for MLP–Back + SMOTE
MLP–Grad + SMOTE vs. MLP–Grad	873.0	117.0	0.000	Rejected for MLP–Grad + SMOTE
RBFN–Decr + SMOTE vs. RBFN–Decr	816.0	174.0	0.000	Rejected for RBFN–Decr + SMOTE
RBFN–Incr + SMOTE vs. RBFN–Incr	811.0	179.0	0.000	Rejected for RBFN–Incr + SMOTE

The null-hypothesis of equality is rejected in all cases, which supports the conclusion that CO^{2}RBFN outperforms the remaining algorithms when the original data sets are used.

6.2. Analysis of the significance of the pre-processing mechanism: use of SMOTE

In this case, the complete table of results with the application of the SMOTE pre-processing technique is shown in Table 7, which follows the same structure as the previous one. Also in this case the CO^{2}RBFN approach achieves the highest result in test among all the algorithms compared in this analysis.

Now, we will apply a Wilcoxon signed-ranks test to compare the results of each algorithm with SMOTE pre-processing and without pre-processing.

The results obtained are shown in Table 8, from which we can conclude that the use of pre-processing is a necessity in the framework of imbalanced data sets as the results when applying SMOTE improve significantly for all the algorithms used in this paper.

Once we have shown the significance of pre-processing for the algorithms selected for the experimental study, the second part of this study is aimed to analyse the performance among these methods by considering all the data sets which have been pre-processed with SMOTE.

First, Fig. 8 shows the average ranking computed for all approaches according to the GM metric, where we can observe that CO^{2}RBFN is the best algorithm, because it has obtained the lowest value in the ranking.

The Iman and Davenport test obtains a p-value near to zero, which implies that there are significant differences among the results and thus we should apply a post hoc test to detect which methods are outperformed by CO^{2}RBFN, since it is the best ranked method.

A Holm test, shown in Table 9, compares the best ranking method (CO^{2}RBFN) with the remaining methods. There are significant differences between the control algorithm, CO^{2}RBFN, and RBFN–Decr, LVQ, MLP–Back and RBFN–Incr. There are not significant differences between CO^{2}RBFN and MLP–Grad, C4.5 and HFRBCS.

Table 9
Holm test table with SMOTE pre-processing in all imbalanced data sets. CO^{2}RBFN is the control method.

i	Algorithm	z	p	$\alpha \backslash i$	Hypothesis ($\alpha = 0.05$)
7	RBFN–Decr	5.875	4.23E−9	0.0071	Rejected
6	LVQ	5.157	2.509E−7	0.0083	Rejected
5	MLP–Back	2.851	0.004	0.01	Rejected
4	RBFN–Incr	2.655	0.008	0.0125	Rejected
3	MLP–Grad	2.154	0.031	0.0167	No Rejected
2	C4.5	1.871	0.061	0.025	No Rejected
1	HFRBCS	1.719	0.086	0.05	No Rejected

Table 10
Wilcoxon test table with SMOTE pre-processing in all imbalanced data sets.

R^+ CO^{2}RBFN	R^-		p-Value
648.0	C4.5	306.0	0.027
647.05	HFRBCS	343.0	0.076
915.0	LVQ	75.0	0.000
823.0	MLP–Back	167.0	0.000
651.0	MLP–Grad	339.0	0.069
942.0	RBFN–Decr	48.0	0.000
757.5	RBFN–Incr	232.5	0.002

Next, a Wilcoxon test (Table 10) is applied in order to detect significant differences between the behaviour of pairs of algorithms. As can be seen, in this table the null hypothesis is rejected with a low p-value in all cases.

We must point out that the good behaviour of CO^{2}RBFN with imbalanced data sets and with the original data, could be promoted by the definition of the credit assignment in the algorithm. It considers three factors but only one, the error measure (Eq. (8)) in Section 4.3), is influenced by the imbalanced data. Regarding the contribution factor contribution (Eq. (7)) also in Section 4.3), the effect of imbalanced data is smoothed by the q parameter which considers all the RBFs (almost including almost an RBF for the minority class). The last factor, overlapping (Eq. (9)) is independent of the class distribution for the examples and enables the coverage of all the solution space. We think that the performance of

Algorithm	Ranking
C4.5	4.023
CO^{2}RBFN	**3.045**
HFRBCS	3.943
LVQ	5.739
MLP-Back	4.534
MLP-Grad	4.170
RBFN-Decr	6.114
RBFN-Incr	4.432

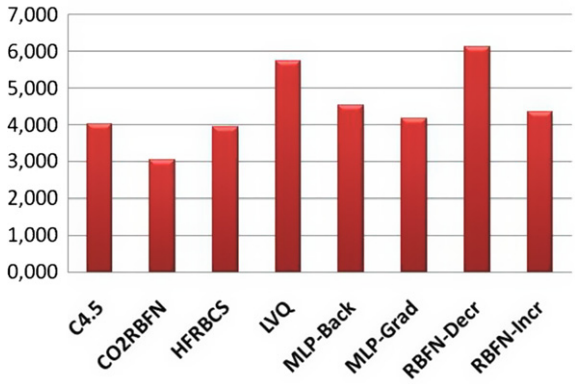

Fig. 8. Ranking according to the GM performance for all algorithms with SMOTE pre-processing. The lower the value, the better.

CO²RBFN with imbalanced data could be increased if an imbalanced precision factor was introduced in the credit assignment.

7. Concluding remarks

In this contribution we have analysed CO²RBFN, a hybrid evolutionary cooperative–competitive algorithm for the design of RBFNs applied to the classification of imbalanced data sets.

The aim of CO²RBFN is to obtain simple and accurate networks and, in this manner, both the design of the general evolutionary paradigm and the rest of the components have this objective. An important key point of CO²RBFN is the identification of the role (credit assignment) of each basis function in the whole network. In order to evaluate this value for a given RBF three factors are defined and used: the RBF contribution to the network's output, a_i (promoting the generality of the RBF); the error in the basis function radius, e_i (reinforcing the quality of the individual); and the degree of overlapping among RBFs, o_i (promoting the good location of the RBFs). In order to drive the cooperative–competitive process, with an adequate balance between exploration and exploitation, four operators are used: *Remove, Random Mutation, Biased Mutation* (based on clustering) and *Null*. The application of these operators is determined by a fuzzy rule-based system which represents expert knowledge of the RBFN design. The inputs of this system are the three parameters used for credit assignment: a_i, e_i, and o_i.

We have studied the effect of a pre-processing stage on the performance of CO²RBFN by contrasting the results obtained using the original data sets against the ones obtained with the SMOTE algorithm. Furthermore, we have included in our experimental study some well-known soft-computing methods for comparison for both the neural network paradigm and rule induction systems.

The experimentation shows that CO²RBFN outperforms, with significant differences, the remaining methods in a imbalanced data sets framework. Pre-processing promotes better results in CO²RBFN, the same as with the other soft-computing methods. But the combination of CO²RBFN with SMOTE allows to obtain the RBFNs with the best prediction capacity of all the methods considered.

The good performance of CO²RBFN in the scenario of imbalanced data sets is due to the definition used for the credit assignment of individuals. In this way and in order to evaluate an individual only one of the three parameters takes into account the accuracy of the RBF. The other ones are for distributing, exploring and optimizing the RBFs in the data set space definition.

As future work, we will focus our studies on the framework of highly imbalanced data sets, analysing the improvement of CO²RBFN and considering an imbalanced precision factor in the credit assignment.

Acknowledgment

This work had been supported by the Spanish Ministry of Science and Technology under Projects TIN2008-06681-C06-01 and TIN2008-06681-C06-02 and the Andalusian Research Plan TIC-3928.

Appendix A. On the use of non-parametric tests based on rankings

In this paper, we have made use of statistical techniques for the analysis of neural network methods, since they are a necessity in order to provide a correct empirical study (Demšar, 2006; García and Herrera, 2008; García et al., 2009). Specifically, we have employed non-parametric tests, due to the fact that the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, making the statistical analysis lose credibility (Demšar, 2006).

In this appendix we describe the procedures for performing pair and multiple comparisons. Specifically, we have employed a Wilcoxon signed-rank test as a non-parametric statistical procedure for performing pairwise comparisons between two algorithms. For multiple comparison we have used an Iman–Davenport test to detect statistical differences. A Holm post hoc test was used in order to find which algorithms partners' average results were dissimilar. We describe these approaches below. Furthermore, any interested reader can find additional information on the website: http://sci2s.ugr.es/sicidm/, together with the software for applying the statistical tests.

A.1. Pairwise comparisons: Wilcoxon signed-ranks test

This is the analogue of the paired *t*-test in non-parametrical statistical procedures; therefore, it is a pairwise test that aims to detect significant differences between the behaviour of two algorithms.

Let d_i be the difference between the performance scores of the two classifiers on *i*th out of N_{ds} data sets. The differences are ranked according to their absolute values; average ranks are assigned in the case of ties. Let R^+ be the sum of ranks for the data sets on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i>0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i), \tag{A.1}$$

$$R^- = \sum_{d_i<0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i), \tag{A.2}$$

Let T be the smallest of the sums, $T = \min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for N_{ds} degrees of freedom (Table B.12 in Zar, 1999), the null hypothesis of equality of means is rejected.

A.2. Multiple comparisons: Iman–Davenport and Holm post hoc tests

In order to perform a multiple comparison, it is necessary to check whether all the results obtained by the algorithms present any inequality. In the case of finding inequality then we can know, by using a post hoc test, which algorithms partners' average results are dissimilar. Next, we describe the non-parametric tests used.

- The Iman and Davenport test (Sheskin, 2006) is a non-parametric test, derived from the Friedman test (Sheskin, 2006):

$$F_F = \frac{(N_{ds}-1)\chi_F^2}{N_{ds}(K-1)-\chi_F^2}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N_{ds}-1)$ degrees of freedom. Statistical tables for critical values can be found at (Sheskin, 2006; Zar, 1999).

- The Holm test (Holm, 1979): it is a multiple comparison procedure which can work with a control algorithm and compares it with the remaining methods. The test statistics for comparing the *i*th and *j*th method using this procedure is

$$z = (R_i - R_j) \Big/ \sqrt{\frac{k(k+1)}{6N_{ds}}}.$$

The z value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate level of confidence α.

A Holm test is a step-up procedure that sequentially tests the

hypotheses ordered by their significance. We will denote the ordered p-values by p_1, p_2,..., so that $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_{k-1}$. The Holm test compares each p_i with $\alpha/(k-i)$, starting from the most significant p value. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and we can compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remain hypotheses are retained as well.

References

Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M., Ventura, S., Garrell, J., Otero, J., Romero, C., Bacardit, J., Rivas, V., Fernández, J., Herrera, F., 2009. Keel: A software tool to assess evolutionary algorithms for data mining problems. Soft Comput. 13 (3), 307–318.

Alejo, R., García, V., Sotoca, J., Mollineda, R., Sánchez, J., 2007. Improving the performance of the RBF neural networks trained with imbalanced samples. LNCS 4507, 162–169.

Al-Haddad, L., Morris, C., Boddy, L., 2000. Training radial-basis function neural networks: Effects of training set size and imbalanced training sets. J. Microbiol. Meth. 43, 33–44.

Ampazis, N., Perantonis, S., 2002. Two highly efficient second-order algorithms for training feedforwards networks. IEEE Trans. Neural Networks 13 (3), 1064–1074.

Asuncion, A., Newman, D., 2007. UCI Machine Learning Repository. School of Information and Computer Science, University of California, Irvine <http://www.ics.uci.edu/mlearn/MLRepository.html>.

Bäck, T., Hammel, U., Schwefel, H., 1997. Evolutionary computation: Comments on the history and current state. IEEE Trans. Evol. Comput. 1 (1), 3–17.

Barandela, R., Sánchez, J., García, V., Rangel, E., 2003. Strategies for learning in class imbalance problems. Pattern Recognition 36 (3), 849–851.

Batista, G., Prati, R., Monard, M., 2004. A study of the behaviour of several methods for balancing machine learning training data. SIGKDD Explor. 6 (1), 20–29.

Bezdek, J., Kuncheva, L., 2001. Nearest prototype classifier designs: An experimental study. Internat. J. Intell. Syst. 16 (12), 1445–1473.

Broomhead, D., Lowe, D., 1988. Multivariable functional interpolation and adaptive networks. Complex Syst. 2, 321–355.

Buchtala, O., Klimek, M., Sick, B., 2005. Evolutionary optimization of radial basis function classifiers for data mining applications. IEEE Trans. Systems Man Cybernet. B 35 (5), 928–947.

Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W., 2002. SMOTE: Synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.

Chawla, N., Japkowicz, N., Kolcz, A., 2004. Special issue on learning from imbalanced data sets. SIGKDD Explor. Newsl. 6 (1), 1–6.

Chen, S., Cowan, C., Grant, P., 1991. Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans. Neural Networks 2, 302–309.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.

Domingos, P., 1999. Metacost: A general method for making classifiers cost sensitive. In: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining. pp. 155–164.

Er, M., Chen, W., Wu, S., 2005. High-speed face recognition base on discrete cosine transform and rbf neural networks. IEEE Trans. Neural Networks 16 (3), 679–691.

Estabrooks, A., Jo, T., Japkowicz, N., 2004. A multiple resampling method for learning from imbalanced data-sets. Computational Intelligence 20 (1), 18–36.

Fernández, A., del Jesus, M., Herrera, F., 2009. Hierarchical fuzzy rule based classification system with genetic rule selection for imbalanced data-set. International Journal of Approximate Reasoning 50, 561–577.

García, S., Herrera, F., 2008. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J. Mach. Learn. Res. 9, 2677–2694.

García, V., Mollineda, R., Sánchez, J.S., 2008. On the k-NN performance in a challenging scenario of imbalance and overlapping. Pattern Anal. Appl. 11 (3–4), 269–280.

García, S., Fernández, A., Luengo, J., Herrera, F., 2009. A study of statistical techniques and performance measures for genetics–based machine learning: Accuracy and interpretability. Soft Comput. 13 (10), 959–977.

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. Inform. Sci. 180, 2044–2064.

Goldberg, D., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Golub, G., Van Loan, C., 1996. Matrix Computations, third ed. Hopkins University Press.

Harpham, C., Dawson, C., Brown, M., 2004. A review of genetic algorithms applied to training radial basis function networks. Neural Comput. Appl. 13, 193–201.

He, H., Garcia, E.A., 2009. Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. 21 (9), 1263–1284.

Holland, J., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press.

Holm, S., 1979. A simple sequentially rejective multiple test procedure. Scand. J. Stat. 6, 65–70.

Huang, Y.M., Hung, C.M., Jiau, H.C., 2006. Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. Nonlinear Anal. Real World Appl. 7 (4), 720–747.

Huang, S., Tan, K., Lee, T., 2008. Adaptive neural network algorithm for control design of rigid-link electrically driven robots. Neurocomputing 71 (4–6), 885–894.

Jang, J., Sun, C., 1993. Functional equivalence between radial basis functions and fuzzy inference systems. IEEE Trans. Neural Networks 4, 156–158.

Jin, Y., Sendhoff, B., 2003. Extracting interpretable fuzzy rules from RBF networks. Neural Process. Lett. 17 (2), 149–164.

Kilic, K., Uncu, O., Türksen, I.B., 2007. Comparison of different strategies of utilizing fuzzy clustering in structure identification. Inform. Sci. 177 (23), 5153–5162.

Lacerda, E., Carvalho, A., Braga, A., Ludermir, T., 2005. Evolutionary radial functions for credit assessment. Appl. Intell. 22, 167–181.

Li, B., Peng, J., Chen, Y., Jin, Y., 2006. Classifying unbalanced pattern groups by training neural network. LNCS 3972, 8–13.

Li, M., Tian, J., Chen, F., 2008. Improving multiclass pattern recognition with a co-evolutionary RBFNN. Pattern Recognition Lett. 29 (4), 392–406.

Maglogiannis, I., Sarimveis, H., Kiranoudis, C., Chatziioannou, A., Oikonomou, N., Aidinis, V., 2008. Radial basis function neural netwroks classification for the recognition of idiopathic pulmonary fibrosis in microscopic images. IEEE Trans. Inform. Technol. Biomed. 12 (1), 42–54.

Mandani, E., Assilian, S., 1975. An experiment in linguistic synthesis with a fuzzy logic controller. Internat. J. Man Mach. Stud. 7 (1), 1–13.

Marcos, J., Hornero, R., Álvarez, D., Del Campo, F., López, M., Zamarrón, C., 2008. Radial basis function classifiers to help in the diagnosis of the obstructive sleep apnoea syndrome from nocturnal oximetry. Med. Biol. Eng. Comput. 46, 323–332.

Mazurowski, M., Habas, P., Zurada, J., Lo, J., Baker, J., Tourassi, G., 2008. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. Neural Networks 21 (2–3), 427–436.

Moller, F., 1990. A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6, 525–533.

Moody, J., Darken, C., 1989. Fast learning in networks of locally-tuned processing units. Neural Comput. 1, 281–294.

Murhphey, Y., Guo, H., 2004. Neural learning from umbalanced data. Appl. Intell. 21, 117–128.

Neruda, R., Kudová, P., 2005. Learning methods for radial basis function networks. Future Gener. Comput. Syst. 21 (7), 1131–1142.

Orr, M., 1995. Regularization on the selection of radial basis function centers. Neural Comput. 7, 606–623.

Orriols-Puig, A., Bernadó-Mansilla, E., 2009. Evolutionary rule-based systems for imbalanced data-sets. Soft Comput. 13 (3), 213–225.

Orriols-Puig, A., Bernadó-Mansilla, E., Goldberg, D.E., Sastry, K., Lanzi, P.L., 2009. Facetwise analysis of XCS for problems with class imbalances. IEEE Trans. Evol. Comput. 13, 260–283.

Padmaja, T.M., Dhulipalla, N., Krishna, P.R., Bapi, R.S., Laha, A., 2007. An unbalanced data classification model using hybrid sampling technique for fraud detection. In: Ghosh, A., De, R.K., Pal, S.K. (Eds.), PReMI, Lecture Notes in Computer Science, Vol. 4815. Springer, pp. 341–348.

Padmaja, T., Krishna, P., Bapi, R., 2008. Majority filter-based minority prediction (MFMP): An approach for unbalanced datasets. In: Proc. IEEE Region 10th Annu. Internat. Conf. TENCON, No. 4766705.

Park, J., Sandberg, I., 1991. Universal approximation using radial-basis function networks. Neural Comput. 3, 246–257.

Park, J., Sandberg, I., 1993. Universal approximation and radial basis function network. Neural Comput. 5 (2), 305–316.

Pedrycz, W., 1998. Conditional fuzzy clustering in the design of radial basis function neural networks. IEEE Trans. Neural Networks 9 (4), 601–612.

Peng, X., King, I., 2008. Robust BMPM training based on second-order cone programming and its application in medical diagnosis. Neural Networks 21 (2–3), 450–457.

Peng, J., Li, k., Huang, D., 2006. A hybrid forward algorithm for RBF neural network construction. IEEE Trans. Neural Networks 17 (6), 1439–1451.

Pérez-Godoy, M., Rivera, A., del Jesus, M., Berlanga, F., 2010. CO^{2}RBFN: An evolutionary cooperative–competitive RBFN design algorithm for classification problems. Soft Comput. 14 (9), 953–971.

Plat, J., 1991. A resource allocating network for function interpolation. Neural Comput. 3 (2), 213–225.

Potter, M., De Jong, K., 2000. Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evol. Comput. 8 (1), 1–29.

Quilan, J., 1993. C4.5: Programs for Machine Learning. Morgan Kauffman Publishers, San Mateo, CA.

Rivas, V., Merelo, J., Castillo, P., Arenas, M., Castellano, J., 2004. Evolving RBF neural networks for time-series forecasting with EVRBF. Inform. Sci. 165, 207–220.

Rivera, A., Rojas, I., Ortega, J., del Jesus, M., 2007. A new hybrid methodology for cooperative–coevolutionary optimization of radial basis function networks. Soft Comput. 11 (7), 655–668.

Rojas, R., Feldman, J., 1996. Neural Networks: A Systematic Introduction. Springer.

Rojas, I., Valenzuela, O., Prieto, A., 1997. Statistical analysis of the main parameters in the definition of radial basis function networks. LNCS 1240, 882–891.

Rui, L., Minghu, J., 2008. Chinese text classification based on the BVB model. In: Proc. 4th Internat. Conf. on Semantics, Knowledge, and Grid, SKG, Vol. 4725942. pp. 376–379.
Sheskin, D., 2006. Handbook of Parametric and Nonparametric Statistical Procedures, second ed. Chapman and Hall/CRC.
Siddiqui, A., Masood, A., Saleem, M., 2009. A locally constrained radial basis function for registration and warping of images. Pattern Recognition Lett. 30 (4), 377–390.
Su, C.-T., Hsiao, Y.-H., 2007. An evaluation of the robustness of MTS for imbalanced data. IEEE Trans. Knowl. Data Eng. 19 (10), 1321–1332.
Su, C.-T., Chen, L.-S., Yih, Y., 2006. Knowledge acquisition through information granulation for imbalanced data. Expert Syst. Appl. 31, 531–541.
Sun, Y., Liang, Y., Zhang, W., Lee, H., Lin, W., Cao, L., 2005. Optimal partition algorithm of the RBF neural network and its application to financial time series forecasting. Neural Comput. Appl. 14 (1), 36–44.
Sun, Y., Kamel, M.S., Wong, A.K., Wang, Y., 2007. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition 40, 3358–3378.
Sun, Y., Wong, A.K.C., Kamel, M.S., 2009. Classification of imbalanced data: A review. Internat. J. Pattern Recognit. Artif. Intell. 23 (4), 687–719.
Suresh, S., Sundararajan, N., Saratchandran, P., 2008. Risk-sensitive loss functions for sparse multi-category classification problems. Inform. Sci. 178 (12), 2621–2638.
Topchy, A., Lebedko, O., Miagkikh, V., Kasabov, N., 1997. Adaptive training of radial basis function networks based on co-operative evolution and evolutionary programming. In: Proc. Internat. Conf. Neural Information Processing (ICONIP), pp. 253–258.
Weiss, G., Provost, F., 2003. Learning when training data are costly: The effect of class distribution on tree induction. J. Artif. Intell. Res. 19, 315–354.
Whitehead, B., Choate, T., 1996. Cooperative–competitive genetic evolution of radial basis function centers and widths for time series prediction. IEEE Trans. Neural Networks 7 (4), 869–880.
Widrow, B., Lehr, M., 1990. 30 Years of adaptive neural networks: Perceptron, madaline and backpropagation. Proc. IEEE 78 (9), 1415–1442.
Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometrics 1, 80–83.
Williams, D., Myers, V., Silvious, M., 2009. Mine classification with imbalanced data. IEEE Geosci. Remot. Sens. Lett. 6 (3), 528–532.
Wilson, D., Martinez, T., 1997. Improved heterogeneous distance functions. J. Artif. Intell. Res. 6 (1), 1–34.
Wu, G., Chang, E., 2005. KBA: Kernel boundary alignment considering imbalanced data distribution. IEEE Trans. Knowl. Data Eng. 17 (6), 786–795.
Wu, S., Chow, T., 2004. Induction machine fault detection using som-based RBF neural networks. IEEE Trans. Ind. Electron. 51 (1), 183–194.
Xu, L., Chow, M., Taylor, L., 2007. Power distribution fault cause identification with imbalanced data using the data mining-based fuzzy classification e-algorithm. IEEE Trans. Power Syst. 22 (1), 164–171.
Yang, Q., Wu, X., 2006. 10 Challenging problems in data mining research. Internat. J. Inform. Technol. Decis. Mak. 5 (4), 597–604.
Zar, J., 1999. Biostatistical Analysis. Prentice Hall, Upper Saddle River, New Jersey.
Zhang, C., Jiang, J., Kamel, M., 2005. Intrusion detection using hierarchical neural networks. Pattern Recognition Lett. 26 (6), 779–791.
Zhao, Z., 2009. A novel modular neural network for imbalanced classification problems. Pattern Recognition Lett. 30 (9), 783–788.
Zhou, Z., Liu, X., 2006. Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans. Knowl. Data Eng. 18 (1), 63–77.