Pattern Recognition 44 (2011) 1821–1833

Contents lists available at ScienceDirect

Pattern Recognition

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

A dynamic over-sampling procedure based on sensitivity for multi-class problems

Francisco Fernández-Navarro *, César Hervás-Martínez, Pedro Antonio Gutiérrez

Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, Albert Einstein Building, 3rd Floor, 14071 Córdoba, Spain

ARTICLE INFO

Article history:
Received 30 December 2009
Received in revised form
16 February 2011
Accepted 18 February 2011
Available online 24 February 2011

Keywords:
Classification
Multi-class
Sensitivity
Accuracy
Memetic algorithm
Imbalanced datasets
Over-sampling method
SMOTE

ABSTRACT

Classification with imbalanced datasets supposes a new challenge for researches in the framework of machine learning. This problem appears when the number of patterns that represents one of the classes of the dataset (usually the concept of interest) is much lower than in the remaining classes. Thus, the learning model must be adapted to this situation, which is very common in real applications. In this paper, a dynamic over-sampling procedure is proposed for improving the classification of imbalanced datasets with more than two classes. This procedure is incorporated into a memetic algorithm (MA) that optimizes radial basis functions neural networks (RBFNNs). To handle class imbalance, the training data are resampled in two stages. In the first stage, an over-sampling procedure is applied to the minority class to balance in part the size of the classes. Then, the MA is run and the data are over-sampled in different generations of the evolution, generating new patterns of the minimum sensitivity class (the class with the worst accuracy for the best RBFNN of the population). The methodology proposed is tested using 13 imbalanced benchmark classification datasets from well-known machine learning problems and one complex problem of microbial growth. It is compared to other neural network methods specifically designed for handling imbalanced data. These methods include different over-sampling procedures in the preprocessing stage, a threshold-moving method where the output threshold is moved toward inexpensive classes and ensembles approaches combining the models obtained with these techniques. The results show that our proposal is able to improve the sensitivity in the generalization set and obtains both a high accuracy level and a good classification level for each class.

1. Introduction

The class imbalance problem occurs when, in a classification problem, there are many more instances of some classes than others. The class imbalance problem is pervasive and ubiquitous, causing trouble to the machine learning community [1,2]. The problem of supervised classification with imbalanced training datasets is addressed in several studies [3,4].

A number of solutions to the class imbalance problem were previously proposed both at the *data* [5] and *algorithmic* levels [6]. The data level approach is usually based on resampling methods, including over-sampling the minority groups (groups of interesting rare examples), or under-sampling the majority groups (groups with large example sizes). The algorithmic level introduces unequal weights for the minority and majority classes in the training strategy to force the classifier to pay more attention to the minority classes. While sampling methods (data level approaches) attempt to balance distributions by considering the representative proportions of class patterns in the distribution, algorithmic level methods consider the costs associated with misclassifying patterns. Instead of creating balanced data distributions through different sampling strategies, algorithmic level methods targets the imbalanced learning problem by using different cost matrices that describe the costs for misclassifying any particular pattern.

In this paper, a dynamic over-sampling procedure is proposed for improving the classification of imbalanced datasets with more than two classes. The base over-sampling procedure is the synthetic minority over-sampling technique (SMOTE) [7]. SMOTE is an over-sampling method, where the minority class is over-sampled by taking each minority class sample and introducing synthetic examples throughout the line segments joining any/all of the k minority class nearest neighbours. Depending upon the amount of over-sampling required, neighbours from the k nearest neighbours are randomly chosen.

This procedure is incorporated into a memetic algorithm (MA) [8] that optimizes radial basis functions neural networks (RBFNNs). The MA combines an evolutionary algorithm (EA) [9],

* Corresponding author. Tel.: +34 957 21 83 49; fax: +34 957 21 83 60.
E-mail address: i22fenaf@uco.es (F. Fernández-Navarro).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.02.019

a clustering process, and a local search (LS) procedure. For this reason, this approach is called memetic radial basis function (MRBF).

We propose two different methodologies which add an over-sampling procedure in the learning algorithm for dealing with imbalanced multi-class datasets: the static smote radial basis function (SSRBF) method and the dynamic smote radial basis function (DSRBF) method. The SSRBF method performs the over-sampling procedure in the preprocessing stage and the DSRBF runs the over-sampling procedure both in the preprocessing stage and in different generations of the MA. The over-sampling procedure duplicates the number of patterns of the minority class in the SSRBF method and the class with the worst accuracy (minimum sensitivity) in the DSRBF method in order to improve sensitivity values of the RBFNNs.

Our proposal contrasts with the previous studies as follows:

- Most previous research uses decision trees as the base classifier (see [10–14]). In the present study, we work with RBFNNs as the classification model.
- Resampling techniques are applied only in the preprocessing stage in most research papers (see [5,15]). In this study, the over-sampling procedure is also performed within the learning process. Therefore, the proposed procedure modifies the structure of the training set (data level) and the classification algorithm (algorithm level).
- Multi-class problems are not reduced to two-class problems ("one-vs-all" or "one-vs-one"), which is a very common heuristic when dealing with multi-class problems [16,17]. By adopting a genuinely multi-class formulation, the complexity of the final model is reduced and, in many cases, better results are obtained [18].

The present paper is organized as follows: a brief analysis of the class imbalance problem is given in Section 2; Section 3 describes the base classifier, the learning algorithm and the over-sampling approaches; Section 4 explains the experiments carried out; and finally, Section 5 summarizes the conclusions of our work.

2. Imbalanced datasets in multi-classification

In this section, we will first introduce the class imbalance problem. Then, we will present the evaluation metrics for this kind of classification problem, and finally we will describe the most popular solutions for the class imbalance problem.

2.1. The class imbalance problem

A dataset is considered to be imbalanced if at least one of the classes (later called a minority class) contains a much smaller number of patterns than the other classes (majority classes). The minority class is usually of primary interest in a given application.

The class imbalance problem has been centre of much attention in research in recent years in the context of data mining (as we can see in the experimental studies of [19] and [20]) because the class imbalance problem is pervasive in a large number of domains of great importance in the data mining community. Some real applications within the scope of the class imbalance problem are: detection of oil spills in satellite radar images [21], detection of fraudulent calls [22], risk management [23], modern manufacturing plants [24], predictive microbiology [25] and text classification [26].

Moreover, the most popular classification modelling systems are reported to be inadequate when faced with the class imbalance problem. These classification systems involve, among others, decision trees [27–29], support vector machines [28,30,31], neural networks [28], Bayesian networks [23], nearest neighbour classifiers [27] and the newly reported associative classification approaches [32]. In our study, as we noted earlier, we will focus the class imbalance problem using RBFNNs models as classification system.

2.2. Evaluation in multi-class imbalanced domains

The imbalanced distribution of classes constitutes a difficulty for standard learning algorithms because they are, in general, biased toward majority classes. As a result, patterns from the majority classes are classified correctly by created classifiers, whereas patterns from the minority class tend to be misclassified.

Classification accuracy may lead to erroneous conclusions since the minority class has very little impact on accuracy as compared to the majority classes [29]. Therefore, accuracy is not an appropriate performance measure with imbalanced datasets.

When there are two classes, an alternative to accuracy to overcome these difficulties is the receiver operating characteristic (ROC) curve [33–35]. The area under the ROC curve (AUC) measures the misclassification rate of one class and the accuracy of the other. The ROC curve and the AUC have been used to enhance the quality of binary classifiers [36–38]. Extension of the standard two class ROC to multi-class problems (J-classes) is attractive, since it would confer the benefits of ROC analysis to more problems in pattern recognition. Recently, different approaches have been proposed to extend ROC analysis for multi-class classification, e.g. [39–41]. These proposals are based on simplifying the multi-class classification problem by using multiple binary classifiers, with the one-vs-one and one-vs-all methods. As an example, Hand and Till [39] proposed an AUC for multi-classification (MAUC) problems for one-vs-one methods, defined as

$$MAUC = \frac{J}{J(J-1)} \sum_{l<k} AUC_{lk} \tag{1}$$

where J is the number of classes. In the most general case, the volume under the ROC surface (VUS) has to be maximized in multi-class classification. The ROC surface can be seen as a Pareto front, where each objective corresponds to one dimension. In those cases where there are more than two classes, then the number of objectives depends on the multi-class method (one-vs-one or one-vs-all method). However, in our approach, a direct multi-class formulation (based on the softmax transformation) has been implemented. Therefore, these metrics are not appropriate to evaluate our classifiers.

In general, a direct multi-class formulation is better for modelling pattern classes than other types of approaches such as one-vs-one or one-vs-all methodologies. Ou and Murphey [18] proved that a direct multi-class formulation provides an optimal decision boundary since the model is trained with the presence of the knowledge of all pattern classes.

In addition to the ROC curve measurement, sensitivity and specificity measures are considered in two-class imbalanced problems (see [16,21]). In [16], the selected metric is the geometric mean of the true rates, which measures the balanced performance of a learning algorithm between these two classes more properly than the accuracy.

In this paper, we will focus on multi-classification problems, where the concepts of true/false positive/negative are not valid. For this reason, the minimum sensitivity (MS) and the correct classification rate or accuracy (C) measures associated with a given classifier g are considered in this work.

Table 1
Confusion matrix of a classifier.

Class	1	2	…	Q	Priors
1	n_{11}	n_{12}	…	n_{1Q}	f_1
2	n_{21}	n_{22}	…	n_{2Q}	f_2
…	…	…	…	…	…
Q	n_{Q1}	n_{Q2}	…	n_{QQ}	f_Q

First, we have to define the *MS* and *C* measurements which are derived from the contingency or confusion matrix *M*.

$$M = \left\{ n_{ij};\ \sum_{i,j=1}^{J} n_{ij} = N \right\} \tag{2}$$

where J is the number of classes, N is the number of training or testing patterns and n_{ij} represents the number of times the patterns are predicted by classifier g to be in class j when they really belong to class i. The diagonal corresponds to correctly classified patterns and the off-diagonal to mistakes in the classification task, as shown in Table 1.

Let us denote the number of patterns associated with class i by $f_i = \sum_{j=1}^{J} n_{ij}$, $i=1,\dots,J$. Let $S_i = n_{ii}/f_i$ be the number of patterns correctly predicted to be in class i with respect to the total number of patterns in class i (sensitivity for class i). Therefore, the sensitivity for class i estimates the probability of correctly predicting a class i example.

From the above quantities the minimum sensitivity (*MS*) of a classifier g is the minimum value of the sensitivities for each class:

$$MS = \min\{S_i;\ i=1,\dots,J\} \tag{3}$$

The correct classification rate or accuracy (C) is defined as

$$C = (1/N)\sum_{j=1}^{J} n_{jj} \tag{4}$$

that is, the rate of all the correct predictions.

The minimum sensitivity and accuracy measures express two features associated with a classifier: global performance C and the accuracy for the worst classified class S. These measures have been simultaneously taken into account in previous studies [42–44], achieving a good performance for the classification of imbalanced data. In this paper, the application of the dynamic over-sampling techniques improves the sensitivity of the classifier population, without drastically decreasing their global accuracy.

2.3. Possible solutions for the class imbalance problem

Several methods have been proposed to improve classifiers learnt from imbalanced data, for a review see [19,20]. Published solutions to the class imbalance problem can be categorized as data level and algorithm level approaches.

At the algorithm level, solutions try to adapt existing classifier learning algorithms to bias towards the minority class, such as cost-sensitive learning [6] and recognition-based learning [45]. Solutions at the algorithm level being either classifier learning algorithm-dependent or application-dependent are shown to be effective if applied in a certain context. These factors indicate the need for additional research efforts to advance the classification of imbalanced data.

At the data level, the objective is to rebalance the distribution per class by resampling the data space, including over-sampling patterns of the minority class and under-sampling patterns of the majority classes. Sometimes this can involve a combination of the two techniques [5,7]. Obvious shortcomings of the resampling (data level) approaches are: (1) the optimal distribution per class of a training dataset is usually unknown; (2) an ineffective resampling strategy may risk losing information about the majority classes when being under-sampled and over-fitting the minority class when being over-sampled; and (3) extra learning cost for analysing and processing data is unavoidable in most cases.

The SSRBF method is a data level approach. Nevertheless, the DSRBF method could be considered a hybrid solution between data level and algorithm level approaches, since in the stage of preprocessing, the over-sampled class is the minority class (data level), while in the dynamic one, the over-sampled class will be the class with minimum sensitivity (algorithm level).

3. Classification method

3.1. Base classifier

We consider radial basis functions neural networks (RBFNNs) [46–50] with softmax outputs and the standard structure as the base classification model. A scheme of these models is given in Fig. 1, where J is the number of classes and m is the number of hidden nodes or RBFs of the neural net. The inputs of the neural net are represented by the vector $\mathbf{x}$, $f_l(\mathbf{x},\boldsymbol{\theta}_l)$ is the output of the neural net for the l-th class, and, after applying the softmax transformation, these outputs are transformed into probabilities that the pattern $\mathbf{x}$ belong to the corresponding class, $g_l(\mathbf{x},\boldsymbol{\theta}_l)$. Finally, $\boldsymbol{\theta} = (\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_{J-1})$ is the vector including all the parameters of the neural net.

The activation function of the j-th node in the hidden layer is given by

$$B_j(\mathbf{x},\mathbf{w}_j) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_j\|^2}{r_j^2}\right) \tag{5}$$

where $\mathbf{w}_j = (\mathbf{c}_j, r_j)$ is the vector of parameters of the j-th hidden node, $\mathbf{c}_j = (c_{j1},\dots,c_{jk})$ is the centre of this node, r_j is the

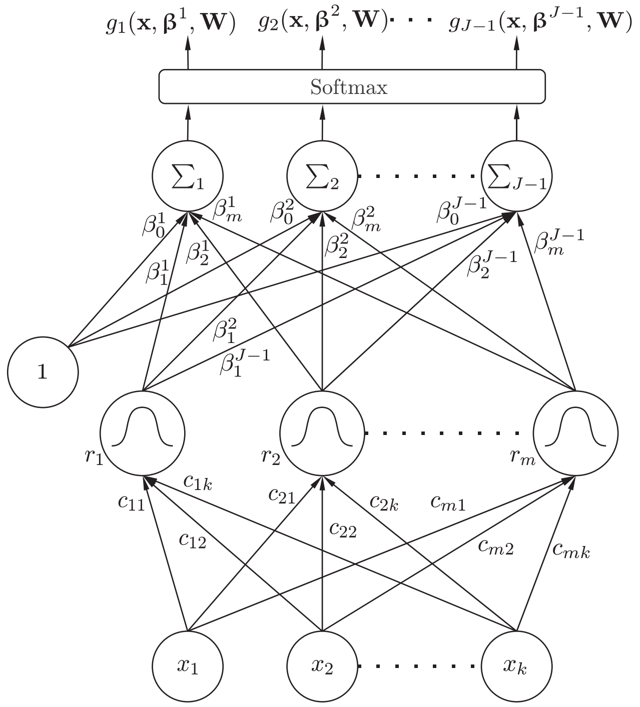
$g_1(\mathbf{x}, \beta^1, \mathbf{W})$
$g_2(\mathbf{x}, \beta^2, \mathbf{W})$
$g_{J-1}(\mathbf{x}, \beta^{J-1}, \mathbf{W})$
Softmax
Σ_1
Σ_2
Σ_{J-1}
β_0^1 β_m^1 β_0^2 β_m^2 β_0^{J-1} β_m^{J-1}
β_1^1 β_2^1 β_2^2 β_2^{J-1} β_1^2 β_1^{J-1}
1
r_1 r_2 r_m
c_{11} c_{1k} c_{12} c_{21} c_{22} c_{2k} c_{m1} c_{m2} c_{mk}
x_1 x_2 x_k

Fig. 1. Structure of radial basis function neural networks: an input layer with k input variables, a hidden layer with m RBFs and an output layer with $J-1$ nodes.

corresponding radium and c_{ji} is the weight of the connection between the i-th input node and the j-th RBF. The activation function of the l-th output node is given by

$$f_l(\mathbf{x},\boldsymbol{\theta}_l)=\beta_0^l+\sum_{j=1}^{m}\beta_j^l B_j(\mathbf{x},\mathbf{w}_j) \tag{6}$$

where $\boldsymbol{\theta}_l=(\beta_0^l,\beta_1^l,\ldots,\beta_m^l,\mathbf{w}_1,\ldots,\mathbf{w}_m)$, β_j^l is the weight of the connection between the j-th RBF and the l-th output node and β_0^l is the bias of the l-th output node. The transfer function of all output nodes is the identity function.

The best RBFNN is determined by means of a memetic algorithm (MA) (detailed in Section 3.2) that optimizes the error function given by the negative log-likelihood for N observations associated with the RBFNN model:

$$L^*(\boldsymbol{\theta})=\frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J-1}y_n^{(l)}f_l(\mathbf{x}_n,\boldsymbol{\theta}_l)+\log\sum_{l=1}^{J-1}\exp f_l(\mathbf{x}_n,\boldsymbol{\theta}_l)\right] \tag{7}$$

where $y_n^{(l)}$ is equal to 1 if the pattern $\mathbf{x}_n$ belongs to the l-th class and equal to 0 otherwise.

3.2. Memetic algorithm

The basic framework of the evolutionary algorithm is the following: the search begins with an initial population of RBFNNs and, in each iteration, the population is updated using a population-update algorithm which evolves both its structure and weights. The population is subject to the operations of replication, mutation and recombination.

In order to overcome the lack of efficiency in the neighbourhood of the global optimum of the EAs, we propose incorporating an optimization method in the evolutionary process. This method is applied at specific stages of the EA.

The MRBF algorithm is detailed in Fig. 2, where p^{B} is the best optimized RBFNN returned by the algorithm.

The main characteristics of the algorithm are the following:

1. *Representation of the individuals.* The algorithm evolves architectures and connection weights simultaneously, each individual being a fully specified RBFNN. RBFNNs are represented using an object-oriented approach and the algorithm deals directly with the RBFNN phenotype. Each connection is specified by a binary value indicating if the connection exists and a real value representing its weights.
2. *Error and fitness functions.* We consider $L^*(\boldsymbol{\theta})$ defined in Eq. (7) as the error function of an individual f of the population.

   The fitness measure needed for evaluating the individuals (Fig. 2, steps 2, 7 and 24) is a strictly decreasing transformation of the error function $L^*(\boldsymbol{\theta})$ given by $A(f)=1/(1+L^*(\boldsymbol{\theta}))$, where $0<A(f)\leq 1$.
3. *Initialization of the population.* The initial population is generated trying to obtain RBFNNs with the maximum possible

```
MRBF Algorithm:
Require: Training dataset ($D$)
Ensure: Best optimized RBFNN ($p^{B}$)
1: $P^{I} \leftarrow \{p_1^{I},\ldots,p_{5000}^{I}\}$ {$p_i^{I}$ is a randomly generated RBFNN}
2: $\forall p_i^{I} \in P^{I}, f_i^{I} \leftarrow A(p_i^{I})$ {Evaluate fitness}
3: $P \leftarrow \{p_{(1)},\ldots,p_{(5000)}\}, (p_{(i)} \prec p_{(j)}) \iff (f_i^{I} > f_j^{I})$ {Sort individuals in $P^{I}$ by increasing
   $f_i^{I}$}
4: $P \leftarrow \{p_{(1)},\ldots,p_{(500)}\}$ {Retain the best 500 RBFNNs}
5: $\forall p_{(i)} \in P, p_{(i)} \leftarrow k-$means($p_{(i)}$) {Improve individuals' centres}
6: while not Stop Condition do
7:     $\forall p_i \in P, f_i \leftarrow A(p_i)$ {Evaluate fitness}
8:     $P \leftarrow \{p_{(1)},\ldots,p_{(500)}\}, (p_{(i)} \prec p_{(j)}) \iff (f_i > f_j)$ {Sort individuals in $P$ by increasing
       $f_i$}
9:     $p^{B} \leftarrow p_{(1)}$ {Store Best Individual}
10:    $P^{P} \leftarrow \{p_{(1)},\ldots,p_{(50)}\}$ {Parametric mutation parents (best 10% of individuals)}
11:    $P^{S} \leftarrow \{p_{(1)},\ldots,p_{(449)}\}$ {Structural mutation parents (best 90% of individuals minusone)}
12:    $P^{R} \leftarrow \{p_{(1)},\ldots,p_{(50)}\}$ {Recombination parents (best 10% of individuals)}
13:    $\forall p_{(i)}^{P} \in P^{P}, p_{(i)}^{P} \leftarrow$ parametric Mutation ($p_{(i)}^{P}$) {Apply parametric mutation}
14:    $\forall p_{(i)}^{S} \in P^{S}, p_{(i)}^{S} \leftarrow$ structural Mutation($p_{(i)}^{S}$) {Apply structural mutation}
15:    $\forall p_{(i)}^{R} \in P^{P}, p_{(i)}^{P} \leftarrow$ recombination ($p_{(i)}^{P}$) {Apply recombination}
16:    $P \leftarrow P^{P} \cup P^{S} \cup P^{R} \cup \{p^{B}\}$ {Offspring including the elite}
17:    if isOptimizationEpoch then
18:        $P^{LS} \leftarrow$ sensitivity Clustering (P) {Apply sensitivity clustering and select closest individu-
           als to centroids}
19:        $P \leftarrow (P - P^{LS})$
20:        $\forall p_{(i)}^{LS} \in P^{LS}, p_{(i)}^{LS} \leftarrow$ local Search ($p_{(i)}^{LS}$) {Apply iRprop + local search}
21:        $P \leftarrow P \cup P^{LS}$ {Include optimized individuals in the population}
22:    end if
23: end while
24: $\forall p_i \in P, f_i \leftarrow A(p_i)$ {Evaluate fitness}
25: $P \leftarrow \{p_{(1)},\ldots,p_{(500)}\}, (p_{(i)} \prec p_{(j)}) \iff (f_i > f_j)$ {Sort individuals in $P$ by increasing $f_i$}
26: $p^{B} \leftarrow p_{(1)}$
27: return $p^{B}$
```

Fig. 2. MRBF training algorithm framework.

fitness. First, 5000 random RBFNNs are generated (Fig. 2, step 1), where the number of RBFs m is a random value in the interval $[M_{\min}, M_{\max}]$. The number of connections between all RBFs of an individual and the input layer is a random value in the interval $[1, k]$ and all of them are connected with the same randomly chosen input variables. In this way, all the RBFs of each individual are initialized in the same random subspace of the input variables. A random value in the $[-I, I]$ interval is assigned for the weights between the input layer and the hidden layer and in the $[-O, O]$ interval for those between the hidden layer and the output layer. The individuals obtained are evaluated using the fitness function and the initial population is finally obtained by selecting the best 500 RBFNNs (Fig. 2, steps 2–4).

In order to improve the randomly generated centres, the standard k-means clustering algorithm [51] is applied using these random centres as the initial centroids for the algorithm and a maximum number of iterations of 100 (Fig. 2, step 5).

4. *Parametric mutation*. Parametric mutation (Fig. 2, step 10) alters the value of the coefficients of the model. Different weight mutations are applied:
   - *Centre and radii mutation*. These parameters are modified in the following way:
     - *Centre creep*. The value of each centre is modified by adding a Gaussian noise, $c_{ji}(t+1)=c_{ji}(t)+\xi(t)$, where $\xi(t)\in N(c_{ji},r_i)$ and $N(c_{ji},r_i)$ represents a one-dimensional normally distributed random variable with mean c_{ji} and with variance the radius of the RBF hidden node.
     - *Radius creep*. The value of each radii is modified by adding another Gaussian noise, $r_i(t+1)=r_i(t)+\xi(t)$, where $\xi(t)\in N(r_i,d)$ and $N(r_i,d)$ represents a one-dimensional normally distributed random variable with mean r_i and with variance the width of the range of each dimension (d).
     - *Randomize centres*. Changes the values of the centres of the hidden neurons to random values ξ where $\xi\in U(0,d)$ and U (0, d) represents a one-dimensional uniform distributed random variable, where d is the width of the range allowed for each dimension of the input space.
     - *Randomize radii*. This operator changes radius values randomly, always with values in the corresponding range of each input space dimension.
   - *Hidden-to-output node connections mutation*. These connections are modified by adding a Gaussian noise, $w(t+1)=w(t)+\xi(t)$, where $\xi(t)\in N(0,T(g))$ and $N(0,T(g))$ represents a one-dimensional normally distributed random variable with mean 0 and with variance the network temperature ($T(g)=1-A\,(g)$) [52].
5. *Structural mutation*. Structural mutation (Fig. 2, step 11) implies a modification in the structure of the RBFNNs and allows the exploration of different regions in the search space, helping to maintain the diversity of the population. There are four different structural mutations: hidden node addition, hidden node deletion, connection addition and connection deletion. These four mutations are applied sequentially to each network. The number of nodes added or deleted in hidden node addition and hidden node deletion is calculated as $\Delta_{\min}+uT(g)[\Delta_{\max}-\Delta_{\min}]$, u being a random uniform variable in the interval [0,1], $T(g)=1-A(g)$ the temperature of the neural net, and $\Delta_{\min}$ and $\Delta_{\max}$ a minimum and maximum number of nodes specified as parameters.

   The connection structural mutations are performed as follows:
   - *Connection addition*. Connection addition mutations are first performed in the hidden layer and then in the output layer. When adding a connection from the input layer to the hidden layer, a node from each layer is selected randomly, and then the connection is added with a random weight. A similar procedure is performed from the hidden layer to the output layer.
   - *Connection deletion*. In the same way, connection deletion mutation is first performed in the hidden layer and then in the output layer, choosing randomly the origin node from the previous layer and the target node from the mutated layer.

   We apply connection mutations sequentially for each mutated neural net, first, adding (or deleting) $1+u[\Delta_o n_o]$ connections from the hidden layer to the output layer and then, adding (or deleting) $1+u[\Delta_h n_h]$ connections from the input layer to the hidden layer, u being a random uniform variable in the interval [0,1], Δ_o and Δ_h previously defined ratios of number of connections in the hidden and the output layer, and n_o and n_h the current number of connections in the output and the hidden layers.

   Parsimony is also encouraged in evolved networks by attempting the four structural mutations sequentially, where node or connection deletion is always attempted before addition. Moreover, the deletion operations are made with higher probability. If a deletion operation is successful, no other mutation will be made. If the probability does not select any mutation, one of the mutations is chosen at random and applied.
6. *Recombination*. Recombination (Fig. 2, step 12) is used to vary the structure of a chromosome or chromosomes from one generation to the next. Different crossover operators are applied :
   - *Binary crossover operator.* This binary operator needs two RBFNNs to be applied. The operator takes an uniformly randomly chosen number of consecutive hidden neurons from the first network, and another random sequence from the second. Then it replaces the first of these sequences by the second one, so that the second individual remains unchanged.
   - *Multipoint crossover operator.* This operator is similar to the previous one, but it replaces with probability 0.2 every hidden neuron of the first RBFNN by a randomly chosen neuron coming from the second net. The second individual remains unchanged again.
7. *Local optimization method*. This method is applied each 50 generations of the evolutionary process and is based on a clustering method that applies the k-means algorithm over a specific space where each individual is mapped to a different point depending on his performance (Fig. 2, step 18). This combination of a clustering process and a local optimization method for EAs was previously proposed [53], reporting good results for regression problems. In this paper, the method has been adapted to classification problems by mapping each individual to the following space: each classifier is represented by the set of the sensitivities of the classifier for each class of the problem. This clustering process is able to obtain groups of individuals that perform similarly for the different classes. After that, we apply *iRprop+* algorithm [54] to the individual closest to the centroid obtained in each cluster (Fig. 2, step 20) and the optimized individuals are returned to the population with their fitness and values updated because our MA is based on the Lamarckian model [55]. The *iRprop+* local improvement procedure is performed considering a maximum of 75 cycles.

3.3. Static over-sampling approach

In this section, the static smote radial basis function (SSRBF) algorithm is described. In this methodology, the dataset is modified only before the MA is performed. The SSRBF algorithm is detailed in Fig. 3, where p^B is the best optimized RBFNN returned by the algorithm.

In the preprocessing stage, the resampling procedure is applied for J steps, where J is the number of classes of the problem (Fig. 3, step 1). In each iteration, the resampling procedure selects the minimum size class, and adds the same number of patterns that it had in the original dataset. Synthetic examples are obtained by applying the SMOTE algorithm over the patterns of the minority class (Fig. 3, step 2).

To determine how many patterns we have to generate and to generate these patterns, there are two possibilities: (1) synthetic patterns previously generated are considered to generate new patterns, (2) only the patterns belonging to the original dataset are taken into account when duplicating the minority class by SMOTE. Experimentally, we observed that the second approach got the best performance, therefore, the over-sampling method selects the class with minimum size and adds the number of patterns that the class had in the original dataset not considering synthetic patterns to generate new samples (Fig. 3, step 3).

Once the resampling procedure has been applied J times, the MRBF algorithm is performed, using as the training set, the dataset generated in the preprocessing stage (Fig. 3, step 5).

Table 2 shows an example for the “zoo” dataset, and it can be seen how the dataset is modified before the MA goes into action.

3.4. Dynamic over-sampling approach

In this section, the dynamic smote radial basis function (DSRBF) algorithm is described. In this approach, the dataset is modified into two stages. First, the dataset is changed before the algorithm performs (taking into account the number of patterns per class) and second, the dataset is increased by adding the number of patterns in the minimum sensitivity class in different generations of the algorithm.

The DSRBF algorithm is detailed in Fig. 4, where p^B is the best optimized RBFNN returned by the algorithm.

The DSRBF method includes a preprocessing stage (Fig. 4, steps 1–4) where the number of minority class patterns is added. The aim will be to decrease the problem imbalance rate by selecting the minority class as the one to which the resampling procedure is applied. Synthetic examples are obtained by the SMOTE algorithm over the patterns of the minority class. This initial preprocessing stage will be applied if the following condition is fulfilled:

$$p^* \leq \frac{1}{2 \cdot J} \tag{8}$$

where J is the number of classes and p^* is the minimum of the estimated prior probabilities (i.e. $p^* = \min\{(f_i/N), 1 \leq i \leq J\}$, where f_i is the number of patterns of the i-th class and N is the total number of patterns). This condition was established since the preprocessing SMOTE should be applied only for the most imbalanced datasets (the size of the minority class is less than a half of the size that this class should have in the ideal balanced case).

After that, the MA runs and every G_q generations from the initial generation G_0, the MA is stopped and the over-sampling procedure proposed is applied. The number of times that the over-sampling procedure is applied is the number of classes of the problem. To test the proposal, the initial over-sampling procedure is conducted in the generation 25 (i.e. $G_0=25$), and then it is repeated every 50 generations ($G_q=50$).

The over-sampling procedure is defined as follows: first, the DSRBF method selects the best RBFNN of the population and determines which class has the minimum sensitivity (Fig. 4, step 9). If two or more classes are classified with the same (minimum) sensitivity, the minority class is selected. The selected class is over-sampled by taking each pattern of the class and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbours. Our implementation currently uses five nearest neighbours as the maximum value of the k parameter in the SMOTE algorithm (Fig. 4, steps 10–14). The over-sampling method adds the number of patterns that the class had in the original dataset not considering synthetic patterns to generate new samples. Once synthetic patterns have been generated, these are inserted into the training set, resulting in the need to re-evaluate and sort the population according to fitness (Fig. 4, steps 15–18).

The preprocessing stage was included in the DSRBF algorithm to improve the convergence of the MA. We observed experimentally that the selected class was the minority one in the first iteration of the dynamic over-sampling procedure, when dealing with very imbalanced datasets. Therefore, we decided to include a preprocessing stage to increase the minority class size in extremely imbalanced datasets, reducing in this way the number of generations of the MA.

Table 2
An example of the static over-sampling approach for the “zoo” dataset.

Iteration	Number of patterns per class	Selected class
Initial	(70, 76, 17, 13, 9, 29)	5
1	(70, 76, 17, 13, 18, 29)	4
2	(70, 76, 17, 26, 18, 29)	3
3	(70, 76, 34, 26, 18, 29)	5
4	(70, 76, 34, 26, 27, 29)	4
5	(70, 76, 34, 39, 27, 29)	5
6	(70, 76, 34, 39, 36, 29)	–

```
SSRBF Algorithm:
Require:  Training dataset (D)
Ensure:   Best optimized RBFNN (pB)
1:  for i = 1, ..., J do
2:      Synthetic Examples ← SMOTE (Minority Class Index, 100, 5) {Minority Class Index is
        the index of the class with the lowest number of patterns in D}
3:      D ← D ∪ Synthetic Examples
4:  end for
5:  Perform theMRBF (D) training algorithm frame work (Figure2)
6:  return p^B
SMOTE (c,p,nn ) generates p% new synthetic samples of the c-th class using the nn nearest neigh-
bours of each pattern.
```

Fig. 3. SSRBF training algorithm framework.

```
DSRBF Algorithm :
Require:  Training data set ($D$)
Ensure:  Best optimized RBFNN ($p^B$)
 1: if $p* < \frac{1}{2 \cdot J}$ then
 2:   Synthetic Examples ← SMOTE (Minority Class Index, 100, 5) {Minority Class Index is
      the index of the class with the lowest number of patterns in $D$}
 3:   $D \leftarrow D \cup$ Synthetic Examples
 4: end if
 5: Perform the initialization process of the MRBF algorithm
 6: while not Stop Condition do
 7:   Perform a base MA iteration of the MRBF algorithm
 8:   if is Over Sampling Epoch then
 9:     [MS Number Of Patterns, MS Class Index] ← MS Class ($p^P$) {Extract the information
        of the class index with minimum sensitivity and its number of patterns}
10:     if MS Number Of Patterns> 5 then
11:       Neighbours ← 5
12:     else
13:       Neighbours ← MS Number Of Patterns −1
14:     end if
15:     Synthetic Examples ← SMOTE( MS Class Index, 100, Neighbours)
16:     $D \leftarrow D \cup$ Synthetic Examples
17:     $\forall p_i \in P, f_i \leftarrow A(p_i)$ {Evaluate fitness}
18:     $P \leftarrow \{p(1),...,p(500)\}, (p(i) \prec p(j)) \Longleftrightarrow (f_i > f_j)$ {Sort individuals in $P$ by increasing
        $f_i$}
19:   end if
20:   $p^B \leftarrow p(1)$
21: end while
22: return $p^B$
SMOTE (c, p, nn) generates p% new synthetic samples of the c-th class using the nn nearest neigh-
bours of each pattern.
```

Fig. 4. DSRBF training algorithm framework.

4. Experiments

The experiments carried out in this section have a double purpose. First of all, the goal is to analyse how the over-sampling techniques affect to the classifier performance. Second, the aim is the justification of the dynamic over-sampling process.

In the first subsection, a description of the datasets and the experimental configuration is given. Then, the main experiments are presented in the subsequent subsections.

4.1. Description of the datasets and the experimental design

The proposed methodologies are applied to 12 datasets taken from the UCI repository [56], to test their overall performance when compared among themselves. One additional dataset described in Section 4.2 has been included, which corresponds to a real predictive microbiology problem of discriminating the growth/no growth of *Staphylococcus aureus*. All the datasets and the corresponding partitions have been included in a public website.[1] Since our method is aimed to improve the accuracy for the worst classified class when dealing with imbalanced datasets, the datasets selected have a considerable imbalance rate.

The selected datasets include five binary problems and eight multi-class problems and present different numbers of instances, features and classes (see Table 3). The CYTvsPOX dataset is the Yeast dataset considering only patterns from CYT and POX classes, and the ME2vsOther is the Yeast dataset considering ME2 patterns vs the remaining patterns. The minimum and maximum number of hidden nodes have been obtained as the best result of a preliminary experimental design ANOVA I, considering a small, medium and high value: $[M_{\min}, M_{\max}] \in \{[1,3],[4,9],[10,12]\}$. This value is also included in Table 3.

The experimental design was conducted using a 10-fold cross-validation, with 10 repetitions per each fold except for the "Saureus4" dataset. The experimental design followed for the Saureus4 dataset is described in Section 4.2. The performance of each method has been evaluated using the correct classification rate (C) and the minimum sensitivity (MS) value for the generalization set, i.e. the accuracy for the worst classified class.

All the parameters used in the evolutionary algorithm, except the maximum and minimum number of RBFs in the hidden layer, have the same values for all problems considered. We have carried out a simple linear rescaling of the input variables in the interval $[-2, 2]$, X_i^* being the transformed variables. The connections between hidden and output layer are initialized in the $[-5, 5]$ interval. The initial value of the radii r_j is obtained in the interval $(0, d_{max}]$, where d_{max} is the maximum distance between two training input examples.

The size of the population is $N=500$. For the structural mutation, the number of nodes that can be added or removed is within the $[1, 2]$ interval, and the number of connections to add or delete in the hidden and the output layers during structural

[1] http://www.uco.es/grupos/ayrna/index.php?lang=en ("Datasets" section).

Table 3
Characteristics of the 13 datasets used for the experiments: number of instances (Size), number of Real (R), binary (B) and nominal (N) input variables, total number of inputs (#In), number of classes (# Out), number of patterns per class (NPPC), minimum and maximum number of hidden nodes used for each dataset ($[M_{\min}, M_{\max}]$) and minimum of the estimated prior probabilities (p^*).

Dataset	Size	R	B	N	In	Out	NPPC	$[M_{\min}, M_{\max}]$	p^*
Hepatitis	155	6	13	–	19	2	(32, 123)	[1, 3]	0.206
BreastC	286	4	3	2	15	2	(201, 85)	[1, 3]	0.297
Haberman	306	3	–	–	3	2	(225, 81)	[1, 3]	0.264
CYTvsPOX	482	8	–	–	8	2	(463, 19)	[4, 9]	0.039
ME2vsOther	1484	8	–	–	8	2	(1433, 51)	[1, 3]	0.034
Newthyroid	215	5	–	–	5	3	(150, 35, 30)	[4, 9]	0.139
Balance	625	4	–	–	4	3	(288, 49, 288)	[4, 9]	0.078
Saureus4	287	3	–	–	3	4	(117, 45, 12, 113)	[4, 9]	0.042
Anneal	898	6	14	18	59	5	(8, 99, 684, 67, 40)	[4, 9]	0.009
Glass	214	9	–	–	9	6	(70, 76, 17, 13, 9, 29)	[9, 12]	0.042
Zoo	101	1	15	–	16	7	(41, 20, 5, 13, 4, 8, 10)	[4, 9]	0.049
E. coli	336	7	–	–	7	8	(143, 77, 52, 35, 20, 5, 2, 2)	[4, 9]	0.006
Yeast	1484	8	–	–	8	10	(463, 429, 30, 163, 51, 44, 35, 244, 20, 5)	[9,12]	0.003

All nominal variables are transformed to binary variables.
BreastC: breast-cancer.

Table 4
Justification of the over-sampling approaches: mean and standard deviation (SD) of the accuracy (C_G (%)) and minimum sensitivity (MS_G (%)) results, mean accuracy ($\overline{C}_G$ (%)), mean minimum sensitivity ($\overline{MS}_G$ (%)) and mean ranking ($\overline{R}$).

	MRBF		SSRBF		DSRBF	
	C_G (%) Mean$_{SD}$	MS_G (%) Mean$_{SD}$	C_G (%) Mean$_{SD}$	MS_G (%) Mean$_{SD}$	C_G (%) Mean$_{SD}$	MS_G (%) Mean$_{SD}$
Hepatitis	$81.82_{8.06}$	$51.00_{18.85}$	$82.84_{6.58}$	$56.31_{20.44}$	**$83.30_{7.44}$**	**$60.50_{21.35}$**
BreastC	**$74.32_{8.20}$**	$47.60_{17.51}$	$62.14_{6.61}$	**$53.23_{10.49}$**	*$67.53_{7.74}$*	*$51.75_{13.72}$*
Haberman	**$73.29_{5.81}$**	$27.12_{11.60}$	$64.77_{8.10}$	**$51.87_{11.42}$**	*$69.48_{6.90}$*	*$44.83_{11.32}$*
CYTvsPOX	*$97.26_{1.96}$*	$47.46_{25.90}$	**$97.33_{1.95}$**	**$56.78_{27.21}$**	$96.49_{3.30}$	*$55.11_{28.63}$*
ME2vsOther	**$97.21_{0.77}$**	$44.43_{15.81}$	*$96.44_{0.34}$*	*$54.86_{4.73}$*	$95.54_{0.59}$	**$66.63_{3.70}$**
Newthyroid	$96.54_{2.71}$	$86.73_{14.42}$	*$97.02_{2.63}$*	*$90.40_{12.69}$*	**$97.60_{2.75}$**	**$91.47_{12.47}$**
Balance	**$95.70_{2.37}$**	$73.58_{21.14}$	$91.75_{4.51}$	*$74.27_{20.01}$*	*$93.85_{3.12}$*	**$79.39_{17.60}$**
Saureus4	**$78.22_{1.56}$**	$8.00_{10.32}$	$72.68_{4.18}$	*$14.00_{9.66}$*	*$72.93_{3.24}$*	**$20.32_{9.41}$**
Anneal	*$98.70_{2.12}$*	$60.52_{42.89}$	$98.55_{3.50}$	*$70.42_{37.89}$*	**$98.82_{2.57}$**	**$77.60_{20.22}$**
Glass	**$68.40_{8.92}$**	$6.97_{17.53}$	*$67.63_{8.65}$*	*$8.63_{18.22}$*	$65.60_{9.28}$	**$16.61_{21.99}$**
Zoo	*$95.33_{6.56}$*	*$62.50_{48.40}$*	$95.13_{6.41}$	$59.00_{49.43}$	**$95.81_{6.53}$**	**$67.00_{47.26}$**
E. coli	*$84.94_{8.59}$*	$26.58_{31.11}$	$84.67_{8.26}$	*$30.02_{31.26}$*	**$85.91_{8.53}$**	**$34.10_{33.75}$**
Yeast	**$58.09_{4.68}$**	*$0.00_{0.00}$*	$52.67_{5.98}$	*$0.00_{0.00}$*	*$55.42_{4.48}$*	**$5.80_{11.12}$**
$\overline{C}_G$ (%) or $\overline{MS}_G$ (%)	**84.60**	41.73	81.81	*47.67*	*82.94*	**51.62**
$\overline{R}$	**1.65**	2.88	2.53	*1.88*	*1.84*	**1.23**

The best result is in bold face and the second best result in italics.

mutations is within the [1, 7] interval. The number of clusters is $k=6$ for the k-mean algorithm of the initialization process. The algorithm stops when 400 generations are completed.

The DSRBF method is compared to different algorithms:

- The MRBF method (detailed in 3.2).
- The SSRBF method (detailed in 3.3).
- Specific neural network methods for imbalanced data proposed in [15]:
  - The over-sampling (OS) algorithm. This method duplicates higher-cost training examples until the appearances of different training examples are proportional to their costs.
  - The SmoteOver-Sampling (SmoteOS) algorithm. Implementation of the SMOTE algorithm in the preprocessing stage to balance in part the datasets. Then, the neural network is trained with the modified dataset.
  - The ThresholdMovNN (TMNN) algorithm. This method moves the output threshold toward inexpensive classes such that examples with higher costs become harder to be misclassified.
  - Hard-ensemble (HE) and soft-ensemble (SE) algorithms. The combination of TMNN, SmoteOS and OS techniques via hard or soft voting schemes.
    These methods have been selected due to their similarities to our model proposed. The first two techniques modify the distribution of the training data such that the costs of the examples are conveyed explicitly by the appearances of the examples. These methods use multilayer perceptron neural networks as the base classifier, and the model is trained by the RProp algorithm.

The MRBF algorithm was implemented in JAVA. For the SSRBF and DSRBF methods, the MRBF algorithm was modified slightly, applying the over-sampling procedures. We also used the cost sensitive neural network (CSNN) Matlab package[2] [15] to obtain the results of the OS, SmoteOS, TMNN, HE and SE methods.

[2] http://lamda.nju.edu.cn/datacode/CSNN.htm

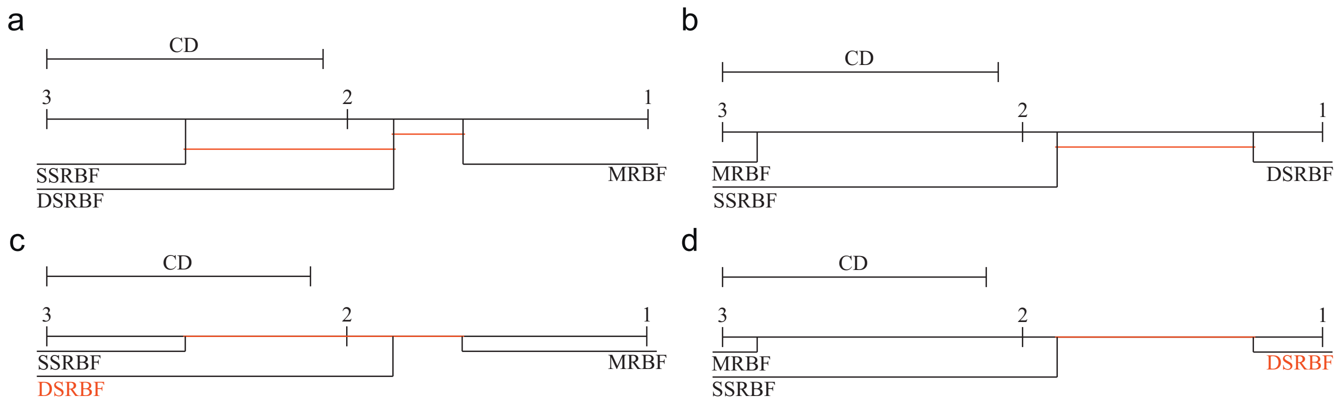

Fig. 5. Justification of the over-sampling approaches: Nemenyi and Bonferroni critical difference diagrams ($\alpha = 0.05$). (a) Nemenyi *(C_G)*, (b) nemenyi *(MS_G)*, (c) bonferroni *(C_G)* and (d) bonferroni *(MS_G)*.

4.2. Description and experimental design of the "Saureus4" dataset

The original dataset was taken from [57,58] describing the growth/no growth boundaries of a five strain cocktail of *Staphylococcus aureus* as a function of temperature, pH and a_w by an ordinary logistic regression model. The same experimental design was followed by carefully choosing a subset (fraction) of the experimental runs of a full factorial design. Data were collected at 8, 10, 13, 16 and 19 °C at pH levels from 4.5 to 7.5 (0.5 intervals) and at 19 levels of a_w (from 0.856 to 0.999 at regular intervals). Four observed microbial responses are obtained based on the probability of microorganism growth [$p=0$ (no growth); $0 < p \leq 0.5$ (probably no growth); $0.5 < p < 1$ (probably growth); $p=1$ (growth)].

The initial dataset (287 conditions) was divided into two parts: training data (146 conditions covering the extreme domain) and generalization data (141 conditions within the interpolation region of the model). The tested algorithms are performed 30 times with the "Saureus4" dataset, since they are stochastic algorithms.

Table 5
Justification of the over-sampling approaches: critical difference (CD) values and differences of rankings of the Bonferroni–Dunn and Nemenyi tests, using DSRBF as the control method.

Method(i)	C_G Method(j)			MS_G Method(j)		
	MRBF	SSRBF	DSRBF	MRBF	SSRBF	DSRBF
Nemenyi test						
MRBF	–	$0.92_{\bullet}$	0.23	–	$1.00^{+}_{\bullet}$	$1.66^{+}_{\bullet}$
SSRBF	–	–	0.69	–	–	0.65
Control method	C_G Compared method			MS_G Compared method		
	MRBF	SSRBF	DSRBF	MRBF	SSRBF	DSRBF
Bonferroni–Dunn test						
DSRBF	0.23	0.69	–	$1.65^{+}_{\bullet}$	0.65	–

Nemenyi test: $CD_{(\alpha=0.1)} = 0.80$, $CD_{(\alpha=0.05)} = 0.91$.
Bonferroni–Dunn test: $CD_{(\alpha=0.1)} = 0.76$, $CD_{(\alpha=0.05)} = 0.87$.
$\bullet$, $\circ$: Statistically difference with $\alpha = 0.05$ ($\bullet$) and $\alpha = 0.1$ ($\circ$).
$+$: The difference is in favour of control method or method(j).

4.3. Justification of the over-sampling approaches

In this subsection, the MRBF method is compared to over-sampling approaches, i.e. the SSRBF and DSRBF methods. The purpose of this section is to show that incorporating the over-sampling procedure in the learning algorithm can improve the performance of classifiers, especially in the class which is more difficult to classify. Table 4 shows the mean and the standard deviation of the correct classification rate (C_G) and minimum sensitivity (MS_G) in the generalization set for each dataset and the MRBF, SSRBF and DSRBF methods. Based on the mean C_G and MS_G, the ranking of each method in each dataset ($R=1$ for the best performing method and $R=3$ for the worst one) is obtained and the mean accuracy and minimum sensitivity ($\overline{C}_G$ and $\overline{MS}_G$) and the mean ranking ($\overline{R}_{C_G}$ and $\overline{R}_{MS_G}$) are also included in Table 4. From the analysis of the results, it can be concluded, from a purely descriptive point of view, that the MRBF method obtained the best results for seven datasets in C_G but the over-sampling techniques achieve the best results for all dataset in MS_G. Furthermore, the MRBF method yields the best mean ($\overline{C}_G = 84.60\%$) and ranking ($\overline{R}_{C_G} = 1.65$) in C_G, however, taking MS_G into account, the DSRBF got the best performance both measures ($\overline{MS}_G = 51.62, \overline{R}_{MS_G} = 1.23$). In general the DSRBF approach substantially improves sensitivity levels (between 5 and 15 points in average) with a slight reduction in accuracy levels for some datasets.

To determine the statistical significance of the rank differences observed for each method in the different datasets, we have carried out a non-parametric Friedman test [59] with the ranking of C_G and MS_G of the best models as the test variables (since a previous evaluation of the C_G and MS_G values results in rejecting the normality and the equality of variances' hypothesis). The test shows that the effect of the method used for classification is statistically significant at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 3.40)$ and the F-distribution statistical values are $F^* = 3.59 \notin C_0$ for C_G and $F^* = 27.18 \notin C_0$ for MS_G. Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking.

Based on this rejection, the Nemenyi post-hoc test is used to compare all classifier to each other. This test considers that the performance of any two classifier is deemed significantly different if their mean ranks differ by at least the critical difference (CD):

$$CD = q\sqrt{\frac{K(K+1)}{6D}} \tag{9}$$

where K and D is the number of classifiers and datasets, and the q value is derived from the studentized range statistic divided by $\sqrt{2}$ [60,61]. However, it has been noted that the approach of comparing all classifiers to each other in a post-hoc test is not as sensitive as the approach of comparing all classifiers to a given classifier (a control method). One approach to this latter type of comparison is the Bonferroni–Dunn test. This test can be computed using Eq. (9) with appropriate adjusted values of q [61].

The results of the Bonferroni–Dunn and Nemenyi tests for $\alpha = 0.10$ and $\alpha = 0.05$ can be seen in Table 5 (and also in the Bonferroni and Nemenyi critical difference diagrams of Fig. 5), using the corresponding critical values. From the results of this test, it can be concluded that DSRBF obtains a significantly better MS_G ranking than the MRBF method. Using C_G as the test variable, the MRBF method does not achieve significantly better results than the DSRBF method. For this reason, the DSRBF methodology is recommended to improve the minimum sensitivity value without a significant loss of accuracy.

4.4. Comparison of the DSRBF method with other recent approaches

In this last subsection, the dynamic over-sampling approach (DSRBF) is compared to other methodologies with a static over-sampling procedure (OS and SmoteOS), to a methodology that uses the original training set to train a neural network, and the cost-sensitivity is introduced in the test phase (TMNN) and to ensembles approaches (SE and HE). The purpose of this section is to show that the DSRBF method is a competitive approach when compared to other state-of-the-art neural network methods specifically designed for dealing with imbalanced datasets.

Table 6
Comparison of the DSRBF method with other recent approaches: mean and standard deviation (SD) of the accuracy (C_G (%)) and minimum sensitivity (MS_G (%)) results, mean accuracy ($\overline{C}_G$ (%)), mean minimum sensitivity ($\overline{MS}_G$ (%)) and mean ranking ($\overline{R}$).

	OS		SmoteOS		TMNN		SE		HE		DSRBF	
	C_G (%) Mean$_{SD}$	MS_G (%) Mean$_{SD}$	C_G (%) Mean$_{SD}$	MS_G (%) Mean$_{SD}$	C_G (%) Mean$_{SD}$	MS_G(%) Mean$_{SD}$	C_G (%) Mean$_{SD}$	MS_G(%) Mean$_{SD}$	C_G (%) Mean$_{SD}$	MS_G (%) Mean$_{SD}$	C_G(%) Mean$_{SD}$	MS_G (%) Mean$_{SD}$
Hepatitis	$\mathit{80.67}_{\mathit{8.61}}$	$47.69_{24.18}$	$80.05_{8.39}$	$51.08_{24.20}$	$79.4_{8.87}$	$46.72_{23.32}$	$80.61_{7.22}$	$48.68_{23.54}$	$80.36_{7.09}$	$\mathit{49.09}_{\mathit{23.29}}$	$\mathbf{83.30}_{\mathbf{7.44}}$	$\mathbf{60.50}_{\mathbf{21.35}}$
BreastC	$61.78_{9.16}$	$38.09_{16.60}$	$63.26_{8.78}$	$\mathit{40.50}_{\mathit{15.88}}$	$63.75_{8.79}$	$34.24_{17.96}$	$64.84_{8.70}$	$34.96_{20.11}$	$\mathit{65.02}_{\mathit{8.22}}$	$34.51_{19.54}$	$\mathbf{67.53}_{\mathbf{7.74}}$	$\mathbf{51.75}_{\mathbf{13.72}}$
Haberman	$61.05_{16.09}$	$28.93_{15.53}$	$61.56_{14.08}$	$31.11_{14.94}$	$\mathit{68.94}_{\mathit{9.23}}$	$27.81_{26.62}$	$64.25_{12.17}$	$30.59_{19.32}$	$63.57_{13.18}$	$\mathit{31.28}_{\mathit{19.11}}$	$\mathbf{69.48}_{\mathbf{6.90}}$	$\mathbf{44.83}_{\mathbf{11.32}}$
CYTvsPOX	$83.22_{15.72}$	$49.85_{25.37}$	$84.41_{14.57}$	$52.80_{24.32}$	$\mathbf{97.53}_{\mathbf{2.05}}$	$50.00_{31.78}$	$90.58_{8.65}$	$\mathit{53.39}_{\mathit{27.44}}$	$84.83_{13.21}$	$52.93_{25,33}$	$\mathit{96.49}_{\mathit{3.30}}$	$\mathbf{55.11}_{\mathbf{28.63}}$
ME2vsOther	$81.28_{10.30}$	$65.95_{12.02}$	$83.83_{9.36}$	$\mathbf{69.25}_{\mathbf{12.28}}$	$\mathbf{96.31}_{\mathbf{1.12}}$	$24.53_{23.91}$	$89.28_{8.04}$	$61.16_{18.96}$	$86.24_{9.62}$	$65.93_{12.56}$	$\mathit{95.54}_{\mathit{0.59}}$	$\mathit{66.63}_{\mathit{3.70}}$
Newthyroid	$97.01_{2.60}$	$87.90_{14.26}$	$97.44_{2.90}$	$\mathit{91.45}_{\mathit{11.53}}$	$96.92_{3.11}$	$88.10_{15.14}$	$\mathit{97.52}_{\mathit{2.68}}$	$90.60_{13.44}$	$97.51_{2.60}$	$90.53_{13.41}$	$\mathbf{97.60}_{\mathbf{2.75}}$	$\mathbf{91.47}_{\mathbf{12.47}}$
Balance	$90.98_{4.01}$	$78.58_{16.72}$	$88.28_{5.22}$	$75.09_{15.85}$	$91.80_{2.68}$	$25.31_{24.94}$	$\mathit{92.29}_{\mathit{3.65}}$	$\mathit{79.16}_{\mathit{16.23}}$	$92.27_{3.47}$	$77.83_{17.23}$	$\mathbf{93.86}_{\mathbf{3.12}}$	$\mathbf{79.39}_{\mathbf{17.60}}$
Saureus4	$63.48_{6.19}$	$19.91_{16.27}$	$62.03_{6.58}$	$\mathbf{21.57}_{\mathbf{8.97}}$	$\mathbf{75.77}_{\mathbf{2.17}}$	$2.49_{6.17}$	$72.70_{3.47}$	$15.13_{8.61}$	$69.83_{3.45}$	$14.38_{10.74}$	$\mathit{72.93}_{\mathit{3.24}}$	$\mathit{20.32}_{\mathit{9.41}}$
Anneal	$92.20_{5.75}$	$56.07_{31.51}$	$91.48_{11.17}$	$66.96_{36.80}$	$95.37_{3.25}$	$37.50_{44.02}$	$\mathit{97.78}_{\mathit{1.41}}$	$\mathit{76.82}_{\mathit{22.50}}$	$96.48_{1.42}$	$73.55_{25.30}$	$\mathbf{98.82}_{\mathbf{2.57}}$	$\mathbf{77.60}_{\mathbf{20.22}}$
Glass	$55.58_{9.73}$	$9.25_{15.63}$	$55.96_{10.26}$	$\mathit{14.18}_{\mathit{18.92}}$	$\mathit{65.49}_{\mathit{8.67}}$	$0.00_{0.00}$	$63.95_{9.18}$	$7.57_{15.58}$	$61.52_{8.99}$	$8.98_{16.78}$	$\mathbf{65.60}_{\mathbf{9.28}}$	$\mathbf{16.61}_{\mathbf{22.00}}$
Zoo	$92.45_{9.10}$	$47.00_{48.63}$	$90.12_{11.42}$	$46.25_{49.03}$	$89.74_{11.15}$	$39.25_{47.44}$	$93.30_{7.66}$	$50.00_{49.75}$	$\mathit{93.40}_{\mathit{7.68}}$	$\mathit{52.00}_{\mathit{49.20}}$	$\mathbf{95.81}_{\mathbf{6.53}}$	$\mathbf{67.00}_{\mathbf{47.26}}$
E. coli	$83.98_{10.47}$	$36.16_{35.74}$	$85.02_{10.40}$	$35.72_{37.15}$	$\mathbf{86.93}_{\mathbf{7.75}}$	$35.18_{35.01}$	$85.60_{9.39}$	$\mathbf{38.41}_{\mathbf{34.50}}$	$85.34_{9.57}$	$\mathit{37.42}_{\mathit{36.18}}$	$\mathit{85.91}_{\mathit{8.53}}$	$34.10_{33.75}$
Yeast	$50.72_{5.38}$	$5.27_{10.27}$	$49.56_{4.87}$	$\mathit{5.75}_{\mathit{9.83}}$	$\mathbf{58.64}_{\mathbf{4.29}}$	$0.00_{0.00}$	$\mathit{57.99}_{\mathit{4.31}}$	$0.67_{4.96}$	$54.16_{4.90}$	$4.68_{10.66}$	$55.42_{4.48}$	$\mathbf{5.80}_{\mathbf{11.12}}$
$\overline{C}_G$ (%) or $\overline{MS}_G$ (%)	76.49	43.89	76.38	*46.28*	*82.04*	31.62	80.82	45.16	79.27	45.62	**82.94**	**51.62**
$\overline{R}$	5.15	4.00	5.23	*2.84*	3.00	5.76	2.73	3.38	3.42	3.46	**1.46**	**1.53**

The best result is in bold face and the second best result in italics.

Table 7
Comparison of the DSRBF method with other recent approaches: critical difference (CD) values and differences of rankings of the Nemenyi and Bonferroni–Dunn tests, using DSRBF as the control method and C_G or MS_G as the test variable.

Method(i)	C_G Method(j)						MS_G Method(j)					
	OS	SmoteOS	TMNN	SE	HE	DSRBF	OS	SmoteOS	TMNN	SE	HE	DSRBF
Nemenyi test												
OS	–	0.07	$2.15^{+}_{\bullet}$	$2.42^{+}_{\bullet}$	1.73	$3.69^{+}_{\bullet}$	–	1.15	1.76	0.65	0.53	$2.46^{+}_{\bullet}$
SmoteOS	–	–	$2.23^{+}_{\bullet}$	$2.50^{+}_{\bullet}$	1.80	$3.76^{+}_{\bullet}$	–	–	2.93	0.53	0.61	1.30
TMNN	–	–	–	0.26	0.42	1.53	–	–	–	$2.38^{+}_{\bullet}$	$2.30^{+}_{\bullet}$	$4.23^{+}_{\bullet}$
SE	–	–	–	–	0.69	1.26	–	–	–	–	0.07	1.84
HE	–	–	–	–	–	1.96	–	–	–	–	–	$1.92^{+}_{\circ}$
Control method	C_G Compared method						MS_G Compared method					
	OS	SmoteOS	TMNN	SE	HE	DSRBF	OS	SmoteOS	TMNN	SE	HE	DSRBF
Bonferroni–Dunn test												
DSRBF	$3.69^{+}_{\bullet}$	$3.76^{+}_{\bullet}$	1.53	1.26	$1.96^{+}_{\bullet}$	–	$2.46^{+}_{\bullet}$	1.30	$4.23^{+}_{\bullet}$	$1.84^{+}_{\circ}$	$1.92^{+}_{\bullet}$	–

Nemenyi test: $CD_{\alpha=0.1} = 1.89$, $CD_{\alpha=0.05} = 2.09$.
Bonferroni–Dunn test: $CD_{\alpha=0.1} = 1.70$, $CD_{\alpha=0.05} = 1.89$.
$\bullet$, $\circ$: Statistically difference with $\alpha = 0.05$ ($\bullet$) and $\alpha = 0.1$ ($\circ$).
+: The difference is in favour of Method(j) (Nemenyi test) or control method (Bonferroni–Dunn test).

In Table 6, the mean and standard deviation of the correct classification rate and the minimum sensitivity in the generalization set (C_G and MS_G) are shown for each dataset. The mean accuracy and minimum sensitivity ($\overline{C}_G$ and $\overline{MS}_G$) and the mean ranking ($\overline{R}_{C_G}$ and $\overline{R}_{MS_G}$) are also included in Table 6.

From the analysis of the results, it can be concluded (taking C_G into account), from a purely descriptive point of view, that the DSRBF method obtains the best result for eight datasets and the TMNN method yields the highest performance for five datasets. Furthermore, the DSRBF method obtains the best mean ranking ($\overline{R}_{C_G}=1.46$), followed by the SE method ($\overline{R}_{C_G}=2.73$), and reports the highest mean accuracy ($\overline{C}_G=82.94\%$), followed by the TMNN method ($\overline{C}_G=82.04\%$).

Using MS_G as the variable test, a descriptive analysis of the results leads to the following remarks: the DSRBF method obtains the best result for 10 out of 13 datasets, the second best results for two other datasets, and the best mean minimum sensitivity and mean ranking ($\overline{C}_G=51.62\%$ and $\overline{R}_{MS_G}=1.53$).

It is necessary again to ascertain if there are differences in the mean ranking of C_G and MS_G, so a procedure similar to that used in the previous subsection has been applied. The non-parametric Friedman test shows that the effect of the method used for

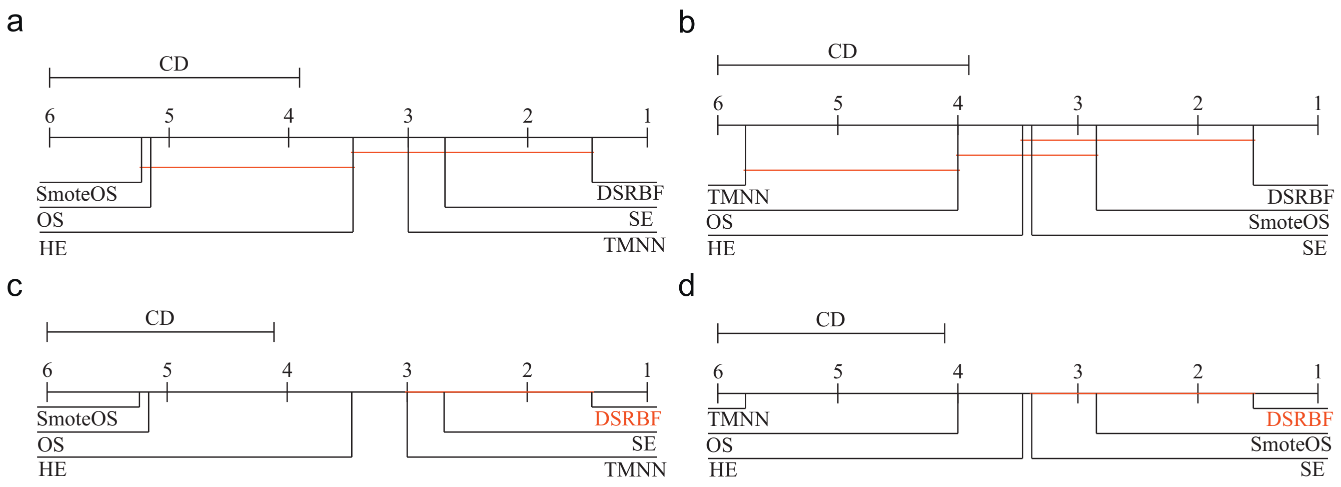

Fig. 6. Comparison of the DSRBF method with other recent approaches: Nemenyi and Bonferroni critical difference diagrams ($\alpha=0.05$). (a) Nemenyi *(C_G)*, (b) nemenyi *(MS_G)*, (c) bonferroni *(C_G)* and (d) bonferroni *(MS_G)*.

Table 8
Probability expression of the best DSRBF model. Performance of this model: accuracy on the training set (C_T), accuracy on the generalization set (C_G), minimum sensitivity (MS) on the training set (MS_T), MS on the generalization set (MS_G), confusion matrix (CM) for the training set (CM_T) and CM for the generalization set (CM_G).

Best DSRBF *S.aureus* probability model

$f_0(\mathbf{x},\boldsymbol{\theta})=-14.85RBF_1-15.00RBF_2-43.88RBF_3+7.85RBF_4-1.45RBF_5+10.75RBF_6+2.22$
$f_1(\mathbf{x},\boldsymbol{\theta})=-5.86RBF_1-9.17RBF_2-3.18RBF_3+9.76RBF_4-6.17RBF_5-0.92$
$f_2(\mathbf{x},\boldsymbol{\theta})=0.46*RBF_1-0.40RBF_2-39.25RBF_3+4.70RBF_4+29.95RBF_5-3.11$
$f_3(\mathbf{x},\boldsymbol{\theta})=0$

$$RBF_0=e^{-0.5\cdot\left(\frac{((a_w^{\star}+2.16)^2)^{0.5}}{1.02}\right)^2}$$

$$RBF_1=e^{-0.5\cdot\left(\frac{((T^{\star}+2.68)^2+(a_w^{\star}+0.48)^2)^{0.5}}{1.67}\right)^2}$$

$$RBF_2=e^{-0.5\cdot\left(\frac{((T^{\star}-0.05)^2+(pH^{\star}+1.83)^2+(a_w^{\star}+1.76)^2)^{0.5}}{1.22}\right)^2}$$

$$RBF_3=e^{-0.5\cdot\left(\frac{((T^{\star}-0.83)^2+(pH^{\star}-1.15)^2+(a_w^{\star}-0.68)^2)^{0.5}}{2.40}\right)^2}$$

$$RBF_4=e^{-0.5\cdot\left(\frac{((pH^{\star}-0.10)^2)^{0.5}}{0.04}\right)^2}$$

$$RBF_5=e^{-0.5\cdot\left(\frac{((T^{\star}-1.55)^2+(pH^{\star}+0.98)^2)^{0.5}}{1.58}\right)^2}$$

$T^{\star},pH^{\star},a_w^{\star}\in[-2,2]$

C_T=78.84%, C_G=81.56%

MS_T=57.14%, MS_G=20.00%

$$CM_T=\begin{pmatrix}50&9&1&0\\6&32&1&5\\0&1&16&11\\0&4&2&51\end{pmatrix};\quad CM_G=\begin{pmatrix}51&5&0&1\\4&15&2&2\\1&1&1&2\\2&3&3&48\end{pmatrix}$$

classification is statistically significant at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 2.36)$ and the F-distribution statistical values are $F^* = 19.03 \notin C_0$ for C_G and $F^* = 14.88 \notin C_0$ for MS_G. Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking.

On the basis of this rejection, the Nemenyi post-hoc test is used to compare all classifier to each other. The differences in rankings between the different algorithms and the results of the Nemenyi test for $\alpha = 0.1$ and 0.05 can be seen in Table 7, using the corresponding critical values. By using this test, it can be seen that the DSRBF method significantly outperforms for $\alpha = 0.05$, the OS and SmoteOS methods using C_G as the test variable and outperforms OS and TMNN methods using MS_G as the test variable for $\alpha = 0.05$, and HE for $\alpha = 0.10$.

The results of the Bonferroni–Dunn test for $\alpha = 0.1$ and 0.05 can be seen in Table 7 (and also in the Bonferroni and Nemenyi critical difference diagrams of Fig. 6) using the corresponding critical values for the two-tailed Bonferroni–Dunn test. From the results of these tests, it can be concluded that the DSRBF method obtains a significantly higher ranking of MS_G when compared to all methods except SmoteOS for $\alpha = 0.10$, which justifies the proposal.

4.5. Analysis of the best DSRBF model obtained for the real microbial growth problem

One of the major advantages of the DSRBF model is the reduced number of features and RBFs included in the final expression, since the MA reduces its complexity by pruning mutations. This can result in a better interpretability of the model, which is especially important when dealing with real problems. In this way, Table 8 includes the best predictor functions of the DSRBF model obtained for the Saureus4 problem. As discussed in Section 4.2, the dataset includes temperature, pH and water activity as input variables and the observations are to be classified in four classes (growth / probably growth / probably no growth / no growth). From these predictor functions, the probability that each pattern x has of belonging to each class can be easily derived by using the softmax functions.

5. Conclusions

This paper addressed the multi-class imbalance problem, combining sampling methods into radial basis function neural networks (RBFNNs) optimized by a memetic algorithm (MA). Two kinds of over-sampling methods were proposed. The first is the static smote radial basis function (SSRBF) method, and the second is the dynamic smote radial basis function (DSRBF) method. The SSRBF method performs the over-sampling procedure in the preprocessing stage and the DSRBF runs the over-sampling procedure both in the preprocessing stage and in different generations of the MA. The proposed methodologies were applied to 12 benchmark classification problems and one real classification problem of microbial growth. It is important to note that all the classification problem considered were chosen within the scope of the imbalanced multi-classification problems.

The results obtained confirm that the dynamic over-sampling approach obtains promising results, achieving excellent mean sensitivity values in almost all datasets analysed, with accuracy values similar to those obtained when the MA is used without over-sampling procedures. However, the improvement in sensitivity implies a loss in accuracy for some imbalanced datasets and the expert has to decide where to apply the methodology or not. The approach has been compared with other neural network learning algorithms specifically designed for dealing with imbalanced datasets, and the results obtained show the competitiveness of the proposal.

Acknowledgements

The authors would like to thank the anonymous reviewers and the editor for comments that helped them to improve this paper. This work has been partially subsidized by the TIN 2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds, and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain). The research of Francisco Fernández-Navarro has been funded by the "Junta de Andalucia" Predoctoral Programme, Grant reference 390015-P08-TIC-3745.

References

[1] N.V. Chawla, N. Japlowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, Aigkdd Explorations 6 (1) (2006) 1–6.
[2] J.H. Zhao, X. Li, Z.Y. Dong, Online rare events detection, in: PAKDD'07, Springer-Verlag, Berlin, Heidelberg, 2007.
[3] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1263–1284.
[4] Y. Sun, A.K.C. Wong, M.S. Kamel, Classification of imbalanced data: a review, International Journal of Pattern Recognition and Artificial Intelligence 23 (4) (2009) 687–719.
[5] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: one-sided selection, in: Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 179–186.
[6] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, C. Brunk, Reducing misclassification costs: knowledge intensive approaches to learning from noisy data, in: Proceedings of the 11th International Conference on Machine Learning (ICML-1994), 1994, pp. 100–109.
[7] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.
[8] P. Moscato, C. Cotta, A gentle introduction to memetic algorithms, Handbook of Metaheuristics, International Series in Operations Research and Management Science, vol. 57, Springer, New York, 2003, pp. 105–144.
[9] T. Back, Evolutionary Algorithms in Theory and Practice, Oxford, 1996.
[10] S. García, F. Herrera, Evolutionary training set selection to optimize c4.5 in imbalanced problems, in: International Conference on Hybrid Intelligent Systems, IEEE Computer Society, 2008, pp. 567–572.
[11] A. Folleco, T.M. Khoshgoftaar, A. Napolitano, Comparison of four performance metrics for evaluating sampling techniques for low quality class-imbalanced data, in: Seventh International Conference on Machine Learning and Applications, ICMLA '08, 2008, pp. 153–158.
[12] L.M. Taft, R.S. Evans, C.R. Shyu, M.J. Egger, N. Chawla, J.A. Mitchell, S.N. Thornton, B. Bray, M. Varner, Countering imbalanced datasets to improve adverse drug event predictive models in labor and delivery, Journal of Biomedical Informatics 42 (2) (2009) 356–364.
[13] A. Orriols-Puig, E. Bernadó-Mansilla, Evolutionary rule-based systems for imbalanced data sets, Soft Computing 13 (3) (2009) 213–225.
[14] J. Stefanowski, S. Wilk, Extending rule-based classifiers to improve recognition of imbalanced classes, in: Studies in Computational Intelligence, vol. 223, 2009.
[15] Z.-H. Zhou, X.-Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Transactions on Knowledge and Data Engineering 18 (1) (2006) 63–77.
[16] A. Fernández, M.J. Del Jesus, F. Herrera, On the influence of an adaptative inference system in fuzzy rule based classification systems for imbalanced data-sets, Expert Systems with Applications 36 (2009) 9805–9812.
[17] J. Xue, D.M. Titterington, Do unbalanced data have a negative effect on lda?, Pattern Recognition 41 (5) (2008) 1575–1588.
[18] G.B. Ou, Y.L. Murphey, Multi-class pattern classification using neural networks, Pattern Recognition 40 (1) (2007) 4–18.
[19] J. Van Hulse, T.M. Khoshgoftaar, A. Napolitano, Experimental perspectives on learning from imbalanced data, Proceedings of the 24th International Conference on Machine Learning (ICML'07), vol. 227, 2007, pp. 935–942.
[20] R.C. Prati, G.E.A.P.A. Batista, M.C. Monard, Class imbalances versus class overlapping: an analysis of a learning system behavior, MICAI 2004: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 2972, 2004, pp. 312–321.
[21] M. Kubat, R.C. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Machine Learning 30 (2–3) (1998) 195–215.
[22] T. Fawcett, F. Provost, Adaptive fraud detection, Data Mining and Knowledge Discovery 1 (3) (1997) 291–316.
[23] K.J. Ezawa, M. Singh, S.W. Norton, Learning goal oriented Bayesian networks for telecommunications management, in: Proceedings of the 13th International Conference on Machine Learning, 1996, pp. 139–147.

[24] P. Riddle, R. Segal, O. Etzioni, Representation design and brute-force induction in a boeing manufacturing domain, Applied Artificial Intelligence 8 (1) (1994) 125–147.
[25] F. Fernández-Navarro, A. Valero, C. Hervás-Martínez, P. Gutíerrez, R. García-Gimeno, G. Zurera-Cosano, Development of a multi-classification neural network model to determine the microbial growth/no growth interface, International Journal of Food Microbiology 141 (2010) 203–212.
[26] C. Cardie, N. Howe, Improving minority class prediction using case-specific feature weights, in: Proceedings of the 14th International Conference on Machine Learning, 1997, pp. 57–65.
[27] G.E.A.P.A. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explorations 6 (1) (2004) 20–29.
[28] N. Japkowicz, S. Stephen, The class imbalance problem: a systematic study, Intelligent Data Analysis Journal 6 (5) (2009) 429–449.
[29] G.M. Weiss, Mining with rarity: a unifying framework, ACM SIGKDD Explorations Newsletter (2004) 67–119.
[30] R. Akbani, S. Kwek, N. Japkowicz, Applying support vector machines to imbalanced datasets, in: Proceedings of the 15th European Conference on Machine Learning (ECML), 2004, pp. 39–50.
[31] B. Raskutti, A. Kowalczyk, Extreme re-balancing for SVMs: a case study, SIGKDD Explorations 6 (1) (2004) 60–69.
[32] A.K.C. Wong, Y. Wang, High-order pattern discovery from discrete-valued data, IEEE Transactions on Knowledge and Data Engineering 9 (6) (1997) 877–893.
[33] F. Provost, T. Fawcett, Analysis and visualization of the classifier performance: comparison under imprecise class and cost distribution, in: Proceedings of the Third International Conference on Knowledge Discovery (KDD97) and Data Mining, AAAI Press, 1997, pp. 43–88.
[34] J. Huang, C.X. Ling, Using auc and accuracy in evaluating learning algorithms, IEEE Transactions on Knowledge and Data Engineering 17 (3) (2005) 299–310.
[35] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27 (2006) 861–874.
[36] H. Mamitsuka, Selecting features in microarray classification using ROC curves, Pattern Recognition 39 (12) (2006) 2393–2404.
[37] M.J. Zolghadri, E.G. Mansoori, Weighting fuzzy classification rules using receiver operating characteristics analysis, Information Sciences 177 (11) (2007) 2296–2307.
[38] C. Marrocco, R.P.W. Duin, F. Tortorella, Maximizing the area under the ROC curve by pairwise feature combination, Pattern Recognition 41 (6) (2008) 1961–1974.
[39] D.J. Hand, R.J. Till, A simple generalisation of the area under the roc curve for multiple class classification problems, Machine Learning 45 (2001) 171–186.
[40] C. Ferri, J. Hernández-Orallo, M.A. Salido, Volume under the roc surface for multi-class problems, Machine Learning: ECML 2003, Lecture Notes in Computer Science, vol. 2837, 2003, pp. 108–120.
[41] R.M. Everson, J.E. Fieldsend, Multi-class roc analysis from a multi-objective optimisation perspective, Pattern Recognition Letters 27 (8) (2006) 918–927.
[42] F.J. Martínez-Estudillo, P.A. Gutiérrez, C. Hervás-Martínez, J.C. Fernández, Evolutionary learning by a sensitivity-accuracy approach for multi-class problems, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC'08), IEEE Press, Hong Kong, China, 2008, pp. 1581–1588.
[43] J. Fernandez, C. Hervas, F. Martinez, P. Gutierrez, Memetic pareto evolutionary artificial neural networks to determine growth/no-growth in predictive microbiology, Applied Soft Computing 11 (1) (2011) 534–550.
[44] J. Fernandez-Caballero, F. Martinez, C. Hervas, P. Gutierrez, Sensitivity versus accuracy in multiclass problems using memetic pareto evolutionary neural networks, IEEE Transactions on Neural Networks 21 (5) (2010) 750–770.
[45] N. Japkowicz, Supervised versus unsupervised binary-learning by feedforward neural networks, Machine Learning 42 (1–2) (2001) 97–122.
[46] J.A.S. Freeman, D. Saad, Learning and generalization in radial basis function networks, Neural Computation 7 (5) (1995) 1000–1020.
[47] Y. Hwang, S. Bang, An efficient method to construct radial basis function neural network classifier, Neural Networks 10 (8) (1997) 1495–1503.
[48] M.J.L. Orr, Regularisation in the selection of radial basis function centres, Neural Computation 7 (3) (1995) 606–623.
[49] C.R. De Silva, S. Ranganath, L.C. De Silva, Cloud basis function neural network: a modified rbf network architecture for holistic facial expression recognition, Pattern Recognition 41 (4) (2008) 1241–1253.
[50] F. Fernández-Navarro, C. Hervás-Martínez, M. Cruz-Ramírez, P.A. Gutiérrez, A. Valero, Evolutionary *q*-Gaussian radial basis function neural network to determine the microbial growth/no growth interface of *Staphylococcus aureus*, Applied Soft Computing 11 (3) (2011) 3012–3020.
[51] K. Fukunaga, Introduction to Statistical Pattern Recognition, second ed., Academic Press, 1999.
[52] F.J. Martínez-Estudillo, C. Hervás-Martínez, P.A. Gutiérrez, A.C. Martínez-Estudillo, Evolutionary product-unit neural networks classifiers, Neurocomputing 72 (1–2) (2008) 548–561.
[53] A.C. Martínez-Estudillo, C. Hervás-Martínez, F.J. Martínez-Estudillo, N. García-Pedrajas, Hybridization of evolutionary algorithms and local search by means of a clustering method, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 36 (3) (2006) 534–545.
[54] C. Igel, M. Hüsken, Empirical evaluation of the improved Rprop learning algorithms, Neurocomputing 50 (6) (2003) 105–123.
[55] D.L. Whitley, V.S. Gordon, K.E. Mathias, Lamarckian evolution, the Baldwin effect and function optimization, in: Y. Davidor, H.P. Schwefel, R. Männer (Eds.), Parallel Problem Solving from Nature—PPSN III, Springer, Berlin, 1994, pp. 6–15.
[56] A. Asuncion, D. Newman, UCI machine learning repository, ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩, 2007.
[57] A. Valero, F. Pérez-Rodríguez, E. Carrasco, J.M. Fuentes-Alventosa, R.M. García-Gimeno, G. Zurera, Modelling the growth boundaries of *Staphylococcus aureus*: effect of temperature, ph and water activity, International Journal of Food Microbiology 133 (1–2) (2009) 186–194.
[58] F. Fernández-Navarro, A. Valero, C. Hervás-Martínez, P.A. Gutiérrez, R.M. García-Gimeno, G. Zurera-Cosano, Development of a multi-classification neural network model to determine the microbial growth/no growth interface, International Journal of Food Microbiology 141 (2010) 203–212.
[59] M. Friedman, A comparison of alternative tests of significance for the problem of *m* rankings, Annals of Mathematical Statistics 11 (1) (1940) 86–92.
[60] O.J. Dunn, Multiple comparisons among means, Journal of the American Statistical Association 56 (1961) 52–56.
[61] Y. Hochberg, A. Tamhane, Multiple Comparison Procedures, John Wiley & Sons, 1987.

Francisco Fernández Navarro was born in Córdoba, Spain, in 1984. He received the B.S. Degree in computer science from the University of Córdoba, Spain, in 2007. He is currently working toward the Ph.D. Degree at the department of computer science and numerical analysis, in the area of computer science and artificial intelligence. His current interests include neural networks evolutionary computation and hybrid algorithms.

César Hervás Martínez, was born in Cuenca, Spain. He received the B.S. Degree in statistics and operating research from the Universidad Complutense, Spain, in 1978 and the Ph.D. Degree in mathematics from the University of Seville, Seville, Spain, in 1986. He is a professor with the University of Córdoba in the department of computing and numerical analysis, in the area of computer science and artificial intelligence. His current research interests include neural networks, evolutionary computation, and the modelling of natural systems.

Pedro Antonio Gutiérrez was born in Córdoba, Spain, in 1982. He received the B.S. degree in Computer Science from the University of Seville, Spain, in 2006, and the Ph.D. degree in Computer Science and Artificial Intelligence from the University of Granada, Spain, in 2009. He is currently an Assistant Professor in the Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba. His current research interests include neural networks and their applications, evolutionary computation, and hybrid algorithms.