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],
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 oversampling 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 oversampling procedure in the preprocessing stage and the DSRBF runs the oversampling 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

$$\text{MAUC} = \frac{1}{J(J-1)} \sum_{k=1}^{J} \text{AUC}_k$$

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.
patterns are predicted by classifier accuracy for the worst classified class. Features associated with a classifier: global performance that is, the rate of all the correct predictions.

\[ C = \frac{1}{N} \sum_{i=1}^{J} \frac{n_{ij}}{n_i} \]  

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.

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

\[ M = \left\{ n_{ij} : \sum_{j=1}^{J} n_{ij} = N \right\} \]  

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.

<table>
<thead>
<tr>
<th>Class</th>
<th>1</th>
<th>2</th>
<th>…</th>
<th>Q</th>
<th>Priors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( n_{11} )</td>
<td>( n_{12} )</td>
<td>…</td>
<td>( n_{1Q} )</td>
<td>( f_1 )</td>
</tr>
<tr>
<td>2</td>
<td>( n_{21} )</td>
<td>( n_{22} )</td>
<td>…</td>
<td>( n_{2Q} )</td>
<td>( f_2 )</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Q</td>
<td>( n_{Q1} )</td>
<td>( n_{Q2} )</td>
<td>…</td>
<td>( n_{QQ} )</td>
<td>( f_Q )</td>
</tr>
</tbody>
</table>

The minimum sensitivity and accuracy measures express two aspects of the classification rate for class \( i \):

\[ n_{ii} / n_i \]  

where \( n_{ii} \) is 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:

\[ MS = \min(S_i ; i = 1, . . . , J) \]  

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

\[ C = (1/N) \sum_{j=1}^{J} n_{ij} \]  

that is, the rate of all the correct predictions.

The minimum sensitivity and accuracy measures express two features associated with a classifier: global performance 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 \( \mathbf{x} \), \( f_j(x, \theta) \) 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, \( g_j(\mathbf{x}, \theta) \). Finally, \( \theta = (\theta_1, . . . , \theta_{J-1}) \) 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

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

where \( \mathbf{w}_j = (c_j, r_j) \) is the vector of parameters of the \( j \)-th hidden node, \( \mathbf{c}_j = (c_{j1}, . . . , c_{jk}) \) is the centre of this node, \( r_j \) is the radius of this node, \( g_1(x, \beta^1, \mathbf{W}) \) \( g_2(x, \beta^2, \mathbf{W}) \ldots \) \( g_{J-1}(x, \beta^{J-1}, \mathbf{W}) \) are the softmax functions.

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 \( c_j \) 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

\[
f_l(x, h_l) = b_{l0} + \sum_{m=1}^{m} y_n^0 f_l(x_n, h_l) + \log \sum_{m=1}^{m} \exp f_l(x_n, h_l)
\]

where \( h_l = (\theta_0, \theta_1, \ldots, \theta_m^l, \ldots, \theta_m^l) \). \( \beta_j^l \) is the weight of the connection between the \( j \)-th RBF and the \( l \)-th output node and \( b_{l0} \) 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:

\[
L^*(\theta) = \frac{1}{N} \sum_{n=1}^{N} \left[ -\sum_{l=1}^{L} y_n^l f_l(x_n, h_l) + \log \sum_{m=1}^{m} \exp f_l(x_n, h_l) \right]
\]

where \( y_n^l \) is equal to 1 if the pattern \( x_n \) 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 \( L^*(\theta) \) 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 \( L^*(\theta) \) given by \( A(f) = 1/(1 + L^*(\theta)) \), 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. \( P^1 = \{ p_1^1, \ldots, p_{500}^1 \} \) (\( p^1 \) is a randomly generated RBFNN)
2. \( \forall p^1 \in P^1, f_1^1 \leftarrow A(p_1^1) \) (Evaluate fitness)
3. \( P \leftarrow \{ p_1^1, \ldots, p_{500}^1 \} \) \( \langle P_i^1 < P_j^1 \rangle \iff (f_i^1 > f_j^1) \) (Sort individuals in \( P^1 \) by increasing \( f_1^1 \))
4. \( P \leftarrow \{ p_1^1, \ldots, p_{500}^1 \} \) (Retain the best 50 RBFNNs)
5. \( \forall p_i^1 \in P, p_i^1 \leftarrow \hat{k} \)-means (\( P_i^1 \)) (Improve individuals’ centres)
6. **while not** Stop Condition **do**
7. \( \forall p_i \in P, f_i \leftarrow A(p_i) \) (Evaluate fitness)
8. \( P \leftarrow \{ p_1, \ldots, p_{500} \} \) \( \langle P_i < P_j \rangle \iff (f_i > f_j) \) (Sort individuals in \( P \) by increasing \( f_1 \))
9. \( p^b \leftarrow P_{\hat{k}} \) (Store Best Individual)
10. \( P^T \leftarrow \{ p_1, \ldots, p_{20} \} \) (Parametric mutation parents (best 10% of individuals))
11. \( P^S \leftarrow \{ p_1, \ldots, p_{400} \} \) (Structural mutation parents (best 90% of individuals minus one))
12. \( P^R \leftarrow \{ p_1, \ldots, p_{50} \} \) (Recombination parents (best 10% of individuals))
13. \( \forall p_i^T \in P^T, p_i^S \leftarrow \) parametric Mutation (\( p_i^T \)) (Apply parametric mutation)
14. \( \forall p_i^S \in P^S, p_i^R \leftarrow \) structural Mutation (\( p_i^S \)) (Apply structural mutation)
15. \( \forall p_i^R \in P^R, p_i^S \leftarrow \) recombination (\( p_i^R \)) (Apply recombination)
16. \( P \leftarrow P^T \cup P^S \cup P^R \cup \{ p^b \} \) (Offspring including the elite)
17. **if** isOptimizationEpoch **then**
18. \( p^{LS} \leftarrow \text{sensitivity Clustering} (P) \) (Apply sensitivity clustering and select closest individuals to centroids)
19. \( P \leftarrow (P - p^{LS}) \)
20. \( p^{LS}_{i,n} \in P^{LS}, p^L_{i,n} \leftarrow \) local Search (\( p^{LS}_{i,n} \)) (Apply \( (\text{Rand} + \text{local search}) \))
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. \( P \leftarrow \{ p_1, \ldots, p_{500} \} \) \( \langle P_i < P_j \rangle \iff (f_i > f_j) \) (Sort individuals in \( P \) by increasing \( f_1 \))
26. \( p^b \leftarrow P_{\hat{k}} \)
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 $[M_{\text{min}}, M_{\text{max}}]$. 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 $[-l, l]$ interval is assigned for the weights between the input layer and the hidden layer and in the $[-0,0]$ 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, $c_i(t+1) = c_i(t) + \zeta(t)$, where $\zeta(t) \in N(c_i(t),I)$ and $N(c_i(t),I)$ represents a one-dimensional normally distributed random variable with mean $c_i$ and with variance the radius of the RBF hidden node.
  - Radius creep. The value of each radii is modified by adding another Gaussian noise, $r_i(t+1) = r_i(t) + \zeta(t)$, where $\zeta(t) \in N(r_i(t),I)$ and $N(r_i(t),I)$ represents a one-dimensional normally distributed random variable with mean $r_i$ and with variance the width of the range of each dimension ($d_i$).
  - Randomize centres. Changes the values of the centres of the hidden neurons to random values $x$ where $x \in U(0,d)$ 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, $w(t+1) = w(t) + \zeta(t)$, where $\zeta(t) \in N(0,T(g))$ 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 $A_{\text{max}} + u[T(g)], A_{\text{max}} - A_{\text{min}}$, $u$ being a random uniform variable in the interval $[0,1]$, $T(g) = 1 - A(g)$ the temperature of the neural net, and $A_{\text{min}}$ and $A_{\text{max}}$ 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) $1 + u[A_{\text{max}} n_h]$ connections from the hidden layer to the output layer and then, adding (or deleting) $1 + u[A_{\text{max}} n_h]$ connections from the input layer to the hidden layer, $u$ being a random uniform variable in the interval $[0,1]$, $A_{\text{max}}$, $A_{\text{min}}$, $n_h$ and $n_o$ previously defined ratios of number of connections in the hidden and the output layer, 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:

\[
p^* \leq \frac{1}{2 \cdot J}
\]  

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 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. \( G_q = 25 \)), 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>
<thead>
<tr>
<th>Iteration</th>
<th>Number of patterns per class</th>
<th>Selected class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>(70, 76, 17, 13, 9, 29)</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>(70, 76, 17, 13, 18, 29)</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>(70, 76, 17, 26, 18, 29)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>(70, 76, 34, 26, 18, 29)</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>(70, 76, 34, 26, 27, 29)</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>(70, 76, 34, 39, 27, 29)</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>(70, 76, 34, 39, 36, 29)</td>
<td>4</td>
</tr>
</tbody>
</table>

| 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)    | 4              |

SSRBF Algorithm:

Require: Training dataset (\( D \))

Ensure: Best optimized RBFNN (\( p^B \))

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:    \( D \) ← \( D \cup \) Synthetic Examples
4: end for
5: Perform the MRBF (\( D \)) training algorithm framework (Figure 2)
6: return \( p^B \)

SMOTE (\( c, p, mn \)) generates \( p\% \) new synthetic samples of the \( c\)-th class using the \( mn \) nearest neighbours of each pattern.
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 design 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

Table 3 presents the datasets used in the experiments. The last column shows the minimum and maximum number of hidden nodes that were selected as the best result of a preliminary experimental design ANOVA I, considering a small, medium and high value: $\frac{M_{\min}}{C_{138}}, \frac{M_{\max}}{C_{138}}$. 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 $[0, 2]$, $X_i$ being the transformed variables. The connections between hidden and output layer are initialized in the $[0, 5]$ 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

---

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 ([Mmin, Mmax]) and minimum of the estimated prior probabilities (p).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>R</th>
<th>B</th>
<th>N</th>
<th>In</th>
<th>Out</th>
<th>NPPC</th>
<th>[Mmin, Mmax]</th>
<th>p^a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hepatitis</td>
<td>155</td>
<td>6</td>
<td>13</td>
<td>-</td>
<td>19</td>
<td>2</td>
<td>32,123</td>
<td>[1, 3]</td>
<td>0.206</td>
</tr>
<tr>
<td>BreastC</td>
<td>286</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>15</td>
<td>2</td>
<td>201,85</td>
<td>[1, 3]</td>
<td>0.297</td>
</tr>
<tr>
<td>Haberman</td>
<td>306</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>2</td>
<td>225,81</td>
<td>[1, 3]</td>
<td>0.264</td>
</tr>
<tr>
<td>CVTrgPOX</td>
<td>482</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>463,19</td>
<td>[4, 9]</td>
<td>0.039</td>
</tr>
<tr>
<td>ME2vOther</td>
<td>1484</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>1433,51</td>
<td>[1, 3]</td>
<td>0.034</td>
</tr>
<tr>
<td>Newthyroid</td>
<td>215</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>150,35,30</td>
<td>[4, 9]</td>
<td>0.139</td>
</tr>
<tr>
<td>Balance</td>
<td>625</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>3</td>
<td>288,49,288</td>
<td>[4, 9]</td>
<td>0.078</td>
</tr>
<tr>
<td>Saureus4</td>
<td>287</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>117,45,12,113</td>
<td>[4, 9]</td>
<td>0.042</td>
</tr>
<tr>
<td>Annel</td>
<td>898</td>
<td>6</td>
<td>14</td>
<td>18</td>
<td>59</td>
<td>5</td>
<td>8,99,684,67,40</td>
<td>[4, 9]</td>
<td>0.009</td>
</tr>
<tr>
<td>Glass</td>
<td>214</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>6</td>
<td>70,76,17,13,9,29</td>
<td>[9,12]</td>
<td>0.042</td>
</tr>
<tr>
<td>Zoo</td>
<td>101</td>
<td>1</td>
<td>15</td>
<td>-</td>
<td>16</td>
<td>7</td>
<td>41,20,5,13,4,8,10</td>
<td>[4, 9]</td>
<td>0.049</td>
</tr>
<tr>
<td>E. coli</td>
<td>336</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>8</td>
<td>143,77,52,35,20,5,2,2</td>
<td>[4, 9]</td>
<td>0.006</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>10</td>
<td>2</td>
<td>463,429,30,163,51,44,35,244,20,5</td>
<td>[9,12]</td>
<td>0.003</td>
</tr>
</tbody>
</table>

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 (%)) and minimum sensitivity (MSC (%) results, mean accuracy (C ()), mean minimum sensitivity (MSC ())), and mean ranking (R).

<table>
<thead>
<tr>
<th></th>
<th>MRBF</th>
<th>SSRRBF</th>
<th>DSRBF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C (%)</td>
<td>MSC (%)</td>
<td>C (%)</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>81.82±95</td>
<td>51.00±18.65</td>
<td>82.84±58</td>
</tr>
<tr>
<td>BreastC</td>
<td>74.32±20</td>
<td>47.60±17.51</td>
<td>62.10±61</td>
</tr>
<tr>
<td>Haberman</td>
<td>73.29±81</td>
<td>27.12±11.60</td>
<td>64.77±10</td>
</tr>
<tr>
<td>CVTrgPOX</td>
<td>97.26±56</td>
<td>47.40±25.00</td>
<td>97.33±95</td>
</tr>
<tr>
<td>ME2vOther</td>
<td>97.21±77</td>
<td>44.43±15.61</td>
<td>96.44±54</td>
</tr>
<tr>
<td>Newthyroid</td>
<td>96.54±71</td>
<td>86.73±14.42</td>
<td>97.02±13</td>
</tr>
<tr>
<td>Balance</td>
<td>95.70±37</td>
<td>73.58±21.14</td>
<td>91.75±41</td>
</tr>
<tr>
<td>Saureus4</td>
<td>78.22±56</td>
<td>8.00±10.32</td>
<td>72.68±18</td>
</tr>
<tr>
<td>Annel</td>
<td>98.70±12</td>
<td>60.52±24.60</td>
<td>98.55±50</td>
</tr>
<tr>
<td>Glass</td>
<td>68.40±32</td>
<td>6.97±15.61</td>
<td>67.63±15</td>
</tr>
<tr>
<td>Zoo</td>
<td>95.33±55</td>
<td>62.50±24.60</td>
<td>95.13±41</td>
</tr>
<tr>
<td>E. coli</td>
<td>84.94±39</td>
<td>26.58±31.11</td>
<td>84.67±20</td>
</tr>
<tr>
<td>Yeast</td>
<td>58.09±68</td>
<td>0.00±0.00</td>
<td>52.67±58</td>
</tr>
<tr>
<td>C (%)</td>
<td>84.60</td>
<td>41.73</td>
<td>81.81</td>
</tr>
<tr>
<td>R</td>
<td>1.65</td>
<td>2.88</td>
<td>2.53</td>
</tr>
</tbody>
</table>

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 SSRRBF 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 similarity 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 SSRRBF 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
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 *aw*, 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 *aw* (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.

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 (\(M_S_G\)) in the generalization set for each dataset and the MRBF, SSRBF and DSRBF methods. Based on the mean \(C_G\) and \(M_S_G\), 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 (\(\bar{C}_G\) and \(\bar{M}_S_G\)) and the mean ranking (\(R_{\bar{C}_G}\) and \(R_{\bar{M}_S_G}\)) 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 \(M_S_G\). Furthermore, the MRBF method yields the best mean \((\bar{C}_G = 84.60\%\) and ranking \((\bar{R}_{\bar{C}_G} = 1.65)\) in \(C_G\), however, taking \(M_S_G\) into account, the DSRBF got the best performance both measures \((\bar{M}_S_G = 51.62, \bar{R}_{M_S_G} = 1.23)\). 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 \(M_S_G\) of the best models as the test variables (since a previous evaluation of the \(C_G\) and \(M_S_G\) 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 \(C_G = (0,0.05 = 3.40)\) and the F-distribution statistical values are \(F^* = 3.59\bar{C}_G\) for \(C_G\) and \(F^* = 27.18\bar{C}_G\) for \(M_S_G\). 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):

\[
CD = q\sqrt{\frac{(K+1)}{6D}}
\]
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 result using the corresponding critical values. From the results of this test, it can be concluded that DSRBF obtains a significantly better result 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 ($C_c(\%)$) and minimum sensitivity ($MS_c(\%)$) results, mean accuracy ($\bar{C}_c(\%)$), mean minimum sensitivity ($\bar{MS}_c(\%)$) and mean rank ($\bar{R}$).

<table>
<thead>
<tr>
<th>Method(\text{i})</th>
<th>\text{OS}</th>
<th>\text{SmoteOS}</th>
<th>\text{TMNN}</th>
<th>\text{SE}</th>
<th>\text{HE}</th>
<th>\text{DSRBF}</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_c(%)$ Mean$_{\text{DSRB}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$MS_c(%)$ Mean$_{\text{DSRB}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Control method)</td>
<td>\multicolumn{6}{c</td>
<td>}{Nemenyi test}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>80.67 &amp; 61.78 &amp; 61.05 &amp; 83.22 &amp; 81.28 &amp; 97.01 &amp; 90.98 &amp; 63.48 &amp; 92.20 &amp; 55.58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmoteOS</td>
<td>47.69 &amp; 38.09 &amp; 28.33 &amp; 48.85 &amp; 59.52 &amp; 87.31 &amp; 78.18 &amp; 19.18 &amp; 56.07 &amp; 92.55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMNN</td>
<td>51.08 &amp; 63.26 &amp; 31.15 &amp; 52.82 &amp; 69.52 &amp; 94.40 &amp; 88.28 &amp; 62.03 &amp; 60.31 &amp; 55.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SE</td>
<td>48.72 &amp; 40.50 &amp; 71.82 &amp; 55.12 &amp; 69.23 &amp; 82.76 &amp; 75.09 &amp; 51.62 &amp; 66.87 &amp; 65.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HE</td>
<td>51.32 &amp; 34.21 &amp; 53.97 &amp; 53.92 &amp; 50.00 &amp; 81.01 &amp; 25.31 &amp; 15.13 &amp; 37.50 &amp; 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_c(%)$ Mean$_{\text{DSRB}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$MS_c(%)$ Mean$_{\text{DSRB}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 $C_c$ and $MS_c$ as the test variable.

<table>
<thead>
<tr>
<th>Method(\text{i})</th>
<th>\text{OS}</th>
<th>\text{SmoteOS}</th>
<th>\text{TMNN}</th>
<th>\text{SE}</th>
<th>\text{HE}</th>
<th>\text{DSRBF}</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_c(%)$ Method(\text{j})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$MS_c(%)$ Method(\text{j})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Control method)</td>
<td>\multicolumn{6}{c</td>
<td>}{Nemenyi test}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>–</td>
<td>0.07</td>
<td>2.15 &amp; 2.42 &amp; 1.73 &amp; 3.69</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmoteOS</td>
<td>–</td>
<td>–</td>
<td>2.23 &amp; 2.50 &amp; 1.80 &amp; 3.76</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMNN</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.26 &amp; 0.42 &amp; 1.53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SE</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.69 &amp; 1.26</td>
<td></td>
</tr>
<tr>
<td>HE</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.96</td>
</tr>
<tr>
<td>$C_c(%)$ Method(\text{j})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$MS_c(%)$ Method(\text{j})</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Control method)</td>
<td>\multicolumn{6}{c</td>
<td>}{Bonferroni-Dunn test}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSRBF</td>
<td>3.69 &amp; 3.76 &amp; 1.53 &amp; 1.26 &amp; 1.96 &amp; –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nemenyi test: $CD_{0.01} = 1.89$, $CD_{0.06} = 2.09$.

Bonferroni-Dunn test: $CD_{0.01} = 1.70$, $CD_{0.06} = 1.89$.

* - Statistically different with $\alpha = 0.05$ (*), and $\alpha = 0.01$ (**). 
+. - The difference is in favour of Method(\text{i}) (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 \(M_{S_{C}}\)) are shown for each dataset. The mean accuracy and minimum sensitivity (\(C_{G}\) and \(M_{S_{C}}\)) and the mean ranking (\(R_{c_{G}}\) and \(R_{M_{S_{C}}}\)) 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 best mean ranking (\(R_{c_{G}} = 1.46\)), followed by the SE method (\(R_{c_{G}} = 2.73\)), and reports the highest mean accuracy (\(C_{G} = 82.94\%\)), followed by the TMNN method (\(C_{G} = 82.04\%\)).

Using \(M_{S_{C}}\) 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 (\(C_{G} = 51.62\%\) and \(R_{M_{S_{C}}} = 1.53\)).

It is necessary again to ascertain if there are differences in the mean ranking of \(C_{G}\) and \(M_{S_{C}}\), 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 (\(x = 0.05\)]. (a) Nemenyi (\(C_{G}\)), (b) nemenyi (\(M_{S_{C}}\)), (c) bonferroni (\(C_{G}\)) and (d) bonferroni (\(M_{S_{C}}\)).

| 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 (\(M_{S_{T}}\)), MS on the generalization set (\(M_{S_{G}}\)), confusion matrix (CM) for the training set (\(C_{T}\)) and CM for the generalization set (\(C_{G}\)).

<table>
<thead>
<tr>
<th>Best DSRBF</th>
<th>S.aureus probability model</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_{0}(x,\theta))</td>
<td>(-14.85RBF_{1} - 15.00RBF_{2} - 43.88RBF_{3} + 7.85RBF_{4} - 1.45RBF_{5} + 10.75RBF_{6} + 2.22)</td>
</tr>
<tr>
<td>(f_{1}(x,\theta))</td>
<td>(-5.86RBF_{1} - 9.17RBF_{2} - 3.18RBF_{3} + 9.76RBF_{4} - 6.17RBF_{5} - 0.92)</td>
</tr>
<tr>
<td>(f_{2}(x,\theta))</td>
<td>(-0.46RBF_{1} - 0.40RBF_{2} - 39.25RBF_{3} + 4.70RBF_{4} + 29.95RBF_{5} + 3.11)</td>
</tr>
<tr>
<td>(f_{3}(x,\theta))</td>
<td>0</td>
</tr>
<tr>
<td>(RBF_{0})</td>
<td>(e^{-\left(\frac{(\alpha_{c} + 2.16)^{2} \beta_{0.5}}{1.02}\right)^{2}})</td>
</tr>
<tr>
<td>(RBF_{1})</td>
<td>(e^{-\left(\frac{(T^{c} + 2.68)^{2} + (\alpha_{e} + 0.48)^{2} \beta_{0.5}}{1.97}\right)^{2}})</td>
</tr>
<tr>
<td>(RBF_{2})</td>
<td>(e^{-\left(\frac{(T^{e} - 0.05)^{2} + (pH^{c} + 1.83)^{2} + (\alpha_{e} + 1.76)^{2} \beta_{0.5}}{1.22}\right)^{2}})</td>
</tr>
<tr>
<td>(RBF_{3})</td>
<td>(e^{-\left(\frac{(T^{e} - 0.83)^{2} + (pH^{c} - 1.15)^{2} + (\alpha_{e} - 0.68)^{2} \beta_{0.5}}{2.40}\right)^{2}})</td>
</tr>
<tr>
<td>(RBF_{4})</td>
<td>(e^{-\left(\frac{(pH^{e} - 0.10)^{2} \beta_{0.04}}{0.04}\right)^{2}})</td>
</tr>
<tr>
<td>(RBF_{5})</td>
<td>(e^{-\left(\frac{(T^{c} - 1.55)^{2} + (pH^{c} + 0.98)^{2} \beta_{0.5}}{1.58}\right)^{2}})</td>
</tr>
<tr>
<td>(T, pH, \alpha_{c} \in [-2, 2])</td>
<td></td>
</tr>
<tr>
<td>(C_{T} = 78.84%, C_{G} = 81.56%)</td>
<td></td>
</tr>
<tr>
<td>(M_{S_{T}} = 57.14%, M_{S_{G}} = 20.00%)</td>
<td></td>
</tr>
</tbody>
</table>
| \(CM_{T}\) | \[
\begin{pmatrix}
50 & 9 & 1 & 0 \\
6 & 32 & 1 & 5 \\
0 & 1 & 16 & 11 \\
0 & 4 & 2 & 51
\end{pmatrix}
\] |
| \(CM_{G}\) | \[
\begin{pmatrix}
51 & 5 & 0 & 1 \\
4 & 15 & 2 & 2 \\
1 & 1 & 1 & 2 \\
2 & 3 & 3 & 48
\end{pmatrix}
\] |
classification is statistically significant at a significance level of 5%, as the confidence interval is \( C_0 = (0.05, 0.236) \) and the \( F \)-distribution statistical values are \( F^* = 19.03 \) and \( C_0 \). 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 \( C_3 \) as the test variable and outperforms OS and TMNN methods using \( M_3C \) 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 \( M_3C \) 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, \( \text{pH} \) 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 Andalucía” Predoctoral Grant, Program reference 390015-P08-TIC-3745.

References


[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.


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 1985. 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.