Contents lists available at ScienceDirect

Neurocomputing

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

Addressing imbalanced classification with instance generation techniques: IPADE-ID

Victoria López [a,*], Isaac Triguero [a], Cristóbal J. Carmona [b], Salvador García [b], Francisco Herrera [a]

[a] Department of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071 Granada, Spain
[b] Department of Computer Science, University of Jaén, 23071 Jaén, Spain

ARTICLE INFO

Article history:
Received 10 May 2012
Received in revised form
30 November 2012
Accepted 10 January 2013
Available online 26 July 2013

Keywords:
Differential evolution
Instance generation
Nearest neighbor
Decision tree
Imbalanced datasets

ABSTRACT

A wide number of real word applications presents a class distribution where examples belonging to one class heavily outnumber the examples in the other class. This is an arduous situation where standard classification techniques usually decrease their performance, creating a handicap to correctly identify the minority class, which is precisely the case under consideration in these applications.

In this work, we propose the usage of the Iterative Instance Adjustment for Imbalanced Domains (IPADE-ID) algorithm. It is an evolutionary framework, which uses an instance generation technique, designed to face the existing imbalance modifying the original training set. The method, iteratively learns the appropriate number of examples that represent the classes and their particular positioning. The learning process contains three key operations in its design: a customized initialization procedure, an evolutionary optimization of the positioning of the examples and a selection of the most representative examples for each class.

An experimental analysis is carried out with a wide range of highly imbalanced datasets over the proposal and recognized solutions to the problem. The results obtained, which have been contrasted through non-parametric statistical tests, show that our proposal outperforms previously proposed methods.

1. Introduction

Classification with imbalanced datasets is a challenging data mining problem that has attracted a lot of attention in the last years [1,2]. This problem is extremely important since it is predominant in many real-world data mining applications including, but not limited to, medical diagnosis, fraud detection, finances, network intrusion and so on. These applications feature samples from one class which are greatly outnumbered by the samples of the other class. Usually, the minority class is the most interesting class from the learning point of view and implies a higher cost of making errors [3,4].

Imbalanced datasets have become an important difficulty to most classifiers, which assume a nearly balanced class distribution [5]. Standard classifiers are developed to minimize a global measure of error, which is independent of the class distribution and causes a bias towards the majority class, paying less attention to the minority class. Consequently, classifying the minority class is more error prone than classifying the majority class, as a huge portion of errors are concentrated in the minority class [6]. Furthermore, the examples of the minority class can be treated as noise and they might be completely ignored by the classifier.

Numerous approaches have been suggested to tackle the problem of classification with imbalanced datasets [1,2,7]. These approaches are developed at both data and algorithm levels. Solutions at the algorithm level modify existing learning algorithms conducting its operations on the improvement of the learning on the minority class [8,9]. Solutions at the data level, also known as data sampling, try to modify the original class distribution in order to obtain a more or less balanced dataset that can be used to correctly identify each class with standard classifiers [10–12].

The use of instance reduction methods [13], which were originally designed for other preprocessing purposes (speed up, noise tolerance and reduction of storage requirements of learning methods [14]), can also be applied to imbalanced datasets [15,16] as a data level solution that is used to find a balance between the minority and the majority classes. It is important that instance reduction methods adapt their bias to this situation to obtain high performances.

An instance reduction process is devoted to find the best reduced set that represents the original training data with a lesser number of instances. This methodology can be divided into Instance Selection (IS) [13,17,18] and Instance Generation (IG) depending on how it creates the reduced set [19,20]. The former process attempts to choose an appropriate subset of the original

* Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
E-mail addresses: vlopez@decsai.ugr.es (V. López), triguero@decsai.ugr.es (I. Triguero), ccarmona@ujaen.es (C.J. Carmona), sglopez@ujaen.es (S. García), herrera@decsai.ugr.es (F. Herrera).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.01.050

training data, while the latter can also build new artificial instances to better adjust the decision boundaries of the classes. In this manner, the IG process fills some regions in the domain of the problem, which have no representative examples in the original dataset. IS methods have been applied to imbalanced datasets with promising results [15,16,21], however, to the best of our knowledge, IG techniques have not been used yet to deal with imbalanced classification problems.

Following the idea of IG techniques, we propose the usage of the Iterative Instance Adjustment for Imbalanced Domains (IPADE-ID) algorithm to deal with highly imbalanced datasets. IPADE-ID is a method inspired by the IG technique IPADE [22,23], that tries to obtain an adequate synthetic training set from the original training set following an incremental approach to determine the most appropriate number of instances per class. The proposal is based in three fundamental operations: a customized initialization procedure, an evolutionary adjustment of the prototypes and the selection of the most representative examples to define the classes. The initialization procedure should be befitting to the specific learning algorithm used with IPADE-ID.

In this work, we choose the Nearest Neighbor (NN) rule [24] and the C4.5 algorithm [25] as learning methods. In this way, we provide suitable initialization procedures for IPADE-ID that matches these learning approaches. At each step, an optimization procedure, based on an adaptive differential evolution algorithm [26–28], adjusts the positioning of the instances generated up to now, and a selection procedure adds new instances if needed. This selection procedure has been particularly designed to consider the existing imbalanced scenario focusing on the performance of the minority class. This informed and organized combination of techniques, leads us to a hybrid artificial intelligent system [29,30] that is able to cope with imbalanced datasets.

In order to analyze the performance of the proposal, we focus on highly imbalanced binary classification problems, having selected a benchmark of 44 problems from KEEL dataset repository[1] [31]. We will perform our experimental analysis focusing on the precision of the models using the Area Under the ROC curve (AUC) [32]. This study will be carried out using non-parametric statistical tests to check whether there are significant differences among the results [33,34].

The rest of the paper is organized as follows. In Section 2, some background about classification with imbalanced datasets and instance generation techniques is given. Next, Section 3 introduces the proposed approach. Sections 4 and 5 describe the experimental framework used and the analysis of results, respectively. Finally, the conclusions achieved in this work are shown in Section 6.

2. Background

This section purpose is to provide the background information needed to describe our proposal. It is divided in two parts: a description of instance generation techniques (Section 2.1) and an introduction to the problem of classification with imbalanced datasets (Section 2.2).

2.1. Instance generation techniques

This section presents the definition and notation for instance generation techniques.

A formal specification of the instance generation problem is the following: Let $\mathbf{x}_p$ be an example where $\mathbf{x}_p = (x_{p1}, x_{p2}, \ldots, x_{pD}, C_p)$, with $\mathbf{x}_p$ belonging to a class C_i given by C_p and a D-dimensional space in which x_{pj} is the value of the j-th feature of the p-th sample. Then, let us assume that there is a training set TR which consists of n instances $\mathbf{x}_p$ and a test set TS composed of t instances $\mathbf{x}_q$, with C_q unknown.

The original purpose of IG is to obtain an instance generated set (GS), which consists of r, $r < n$, instances $\mathbf{p}_u$ where $\mathbf{p}_u = (p_{u1}, p_{u2}, \ldots, p_{uD}, C_u)$, which are either selected or generated from the examples of TR. The instances of the generated set are determined to efficiently represent the distributions of the classes and to discriminate well when used to classify the training objects.

This methodology, also known as instance abstraction, has been widely studied in the specialized literature focusing on the NN rule [24] as target classifier. These techniques follow multiple mechanisms to generate an appropriate GS, such as interpolations between instances, movements of instances and artificial generation of new data. Using the taxonomy proposed in [20], they can be divided into several families depending on the main heuristic operation: positioning adjustment [35], class re-labeling [36], centroid-based [19] and space-splitting [37].

Among these families of methods, the algorithms that are based on the adjustment of the position of the instances were highlighted as outstanding methods in [20]. This methodology can be viewed as an optimization process of the positioning of the instances [38]. The precursor algorithm of this family is the learning vector quantization proposed by Kohonen [39]. One of the most recent and promising algorithms is the model presented in [22], called IPADE, which follows an incremental addition process of instances to determine which classes need more instances to be represented and their best locations.

More information about instance generation approaches (and instance reduction approaches in general) can be found at the SCI2S thematic public website on *Prototype Reduction in Nearest Neighbor Classification: Prototype Selection and Prototype Generation.*[2]

2.2. Imbalanced datasets in classification

In this section we delimit the context in which this work is content, briefly introducing the problem of imbalanced classification. Then, we will describe which approaches are used to deal with this problem, giving special importance to data level approaches that modify the class distribution. We finish this section describing the evaluation metrics that are used in this specific problem with respect to the most common ones in classification.

2.2.1. The problem of imbalanced datasets

In some classification problems, the number of instances that belong to each class is radically different [1,2]. The problem of classification with imbalanced datasets has acquired much relevance in the last time due to its presence in abundant real-world applications such as medical diagnosis [40], finances [41,42], anomaly detection [43] or bioinformatics [44] just naming some of them. Furthermore, the underrepresented class is usually the most interesting class from the learning point of view incorporating high costs when it is not correctly identified [3,4].

In this paper, we focus on two-class imbalanced datasets, where there is a positive (minority) class, with the lowest number of instances, and a negative (majority) class, with the highest number of instances. Although this class distribution is frequent in real data mining problems, standard classifiers are not usually able to cope with the correct identification of positive samples. Frequently, standard classifiers are biased towards the majority class as they are guided by global performance measures, selecting more general rules that cover as many samples as possible and

[1] http://www.keel.es/datasets.php.

[2] http://sci2s.ugr.es/pr/.

disregarding more specific rules that cover few samples (and mostly describe the minority class). Furthermore, samples from the minority class are often ignored by standard classifiers as they are treated as noise.

To organize the datasets according to the degree of imbalance we use the imbalance ratio (IR) [45], defined as the ratio of the number of instances from the negative class and the positive class. The performance of algorithms is usually more degraded when the imbalance increases because positive examples are more easily forgotten. That situation is critical in highly imbalanced datasets because the number of positive instances in the dataset is negligible and that situation increases the difficulty that most learning algorithms have in detecting positive regions. There is no consensus in the literature about when a dataset is considered highly imbalanced or not. We consider that a dataset presents a high degree of imbalance when its IR is higher than 9 (less than a 10% of instances of the positive class), due to the fact that ignoring the minority class instances by a classifier supposes an error of 0.1 in accuracy, a point where it starts to become acceptable. Figs. 1 and 2 depict two datasets with low imbalance and high imbalance respectively.

Traditionally, the IR is the main hint to identify a set of problems which need to be addressed in a special way. However, there are several data intrinsic characteristics that difficult the learning when they appear together with a skewed class distribution, increasing more the difficulty to correctly solve the problem than in separate contexts. These data intrinsic characteristics include the overlapping between classes [46], lack of representative data [47], small disjuncts [6,48], dataset shift [49], presence of borderline and noisy instances [50] and other issues which have interdependent effects with data distribution (imbalance).

2.2.2. *Addressing classification with imbalanced datasets*

The problem of classification with imbalanced datasets has been solved using different strategies, developed at both data and algorithm levels. The goal of solutions at the data-level is to obtain a more or less balanced class distribution that lets a standard classifier perform in a similar manner as in a balanced scenario. In order to do so, solutions at the data level can be categorized as undersampling methods, oversampling methods or hybrid methods. Undersampling methods [51] create a subset of the original dataset by deleting some of the examples of the negative class; oversampling methods [10] create a superset of the original dataset by replicating some of the examples of the positive class or creating new ones from the original positive class instances; and hybrid methods [11] integrate both approaches into one, deleting some of the examples after the application of the oversampling method in order to remove the induced overfitting.

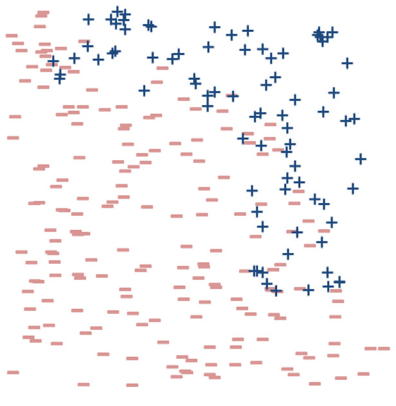
Fig. 1. Dataset with low imbalance ($IR=2.23$).

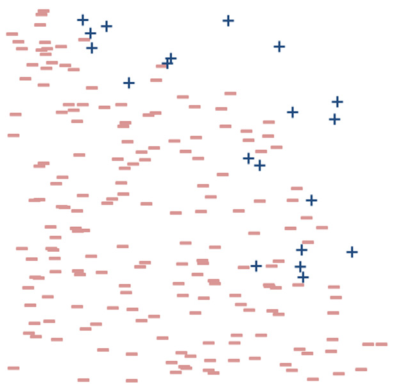
Fig. 2. Dataset with high imbalance ($IR=9.15$).

Solutions at the algorithm level [8,9] try to modify existing learning algorithms in order to bias the learning process towards a correct identification of the positive class instances. Cost-sensitive learning solutions incorporating both the data and algorithmic level approaches assume higher misclassification costs with samples in the positive class and seek to minimize the high cost errors [3,4].

The main advantage of data level solutions is that they are more versatile, since their use is independent of the classifier selected. Furthermore, we may preprocess all datasets beforehand in order to use them to train different classifiers. In this manner, the computation time needed to prepare the data is only required once.

Among the oversampling techniques used to deal with the imbalanced classification problem, the Synthetic Minority Oversampling Technique (SMOTE) [10] algorithm is a well-known reference in the area. In this method the positive class is oversampled by taking each positive class sample and introducing synthetic examples along the line segments joining any/all of the k positive class nearest neighbors.

From the SMOTE algorithm several alternatives have arisen from its way of working. One of the most direct variants of SMOTE is the SMOTE+Edited Nearest Neighbor (SMOTE+ENN) [11], a hybrid data sampling method where SMOTE is applied with the Wilson's ENN rule [52]. This alternative has shown a very robust behavior among many different situations. Borderline-SMOTE [53] is another approach based on the synthetic generation of instances proposed in SMOTE. In this case, only the positive examples that lie near the decision boundaries (the borderline) are used to oversample the positive class. In ADASYN [54], the classification decision boundary is adaptively shifted toward the difficult examples using a weighted distribution for different minority class examples according to their level of difficulty in learning. Safe-Level-SMOTE [55] is another approach that modifies how the SMOTE algorithm works. This approach, modifies the position where the synthetic positive examples are generated, being the new instances closer to the other positive instances than in the original approach.

From the undersampling point of view, we can observe the usage of traditional data cleaning techniques for this purpose. In general, these techniques performance do not correlate their results to the imbalanced area, however, some of them like the Neighborhood Cleaning Rule (NCL) [51] have obtained good results in this scenario [11]. Following this idea several evolutionary algorithms have been used for this purpose [21]. Specifically, we can observe the usage of several proposals with determined algorithms: Evolutionary Sampling [56] is used over C4.5 and Ripper [57], where a search for the best parameters and selection of instances is performed. EUSCHC [15,21,58] was initially proposed to be used with the NN rule, however, later studies demonstrated its effectiveness in rule learners such as C4.5 and PART [59]. Other uses of evolutionary algorithms with undersampling purposes [12] can be done over nested generalized exemplar learning techniques [60] where several significant modifications to the exemplar-based learning model are performed.

In this work we have developed an algorithm that focus its novelty and main features to solve the imbalanced classification problem from the a data level point of view, however, this data level approach modifies its behavior according to the used classifier introducing different operations into its way of working accordingly. In this manner, the current method cannot be viewed as a traditional preprocessing approach for imbalanced classification as it is not independent of the base classified selected and it is not advisable to use the preprocessed datasets before-hand without knowing the learner that will be used in a subsequent stage.

2.2.3. *Evaluation in imbalanced domains*

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

The most used empirical measure, accuracy (1), does not distinguish between the number of correct labels of different classes, which in the ambit of imbalanced problems may lead to erroneous conclusions. For example a classifier that obtains an accuracy of 90% in a dataset with an IR value of 9, might not be accurate if it does not cover correctly any positive class instance.

$$Acc = \frac{TP + TN}{TP + FN + FP + TN} \tag{1}$$

One appropriate metric that could be used to measure the performance of classification over imbalanced datasets is the Receiver Operating Characteristic (ROC) graphics [61]. In these graphics the tradeoff between the benefits and costs can be visualized. They show that any classifier cannot increase the number of true positives without also increasing the false positives. The Area Under the ROC Curve (AUC) [32] corresponds to the probability of correctly identifying which of the two stimuli is noise and which is signal plus noise. AUC provides a single-number summary for the performance of learning algorithms.

The way to build the ROC space is to plot on a two-dimensional chart the true positive rate (Y-axis) against the false positive rate (X-axis) as shown in Fig. 3. The points (0, 0) and (1, 1) are trivial classifiers in which the output class is always predicted as negative and positive, respectively, while the point (0, 1) represents perfect classification. To compute the AUC we just need to obtain the area of the graphic as

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2} \tag{2}$$

where TP_{rate} is the ratio of examples of the positive class that are well-classified and FP_{rate} is the ratio of examples of the negative class misclassified.

Table 1
Confusion matrix for a two-class problem.

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

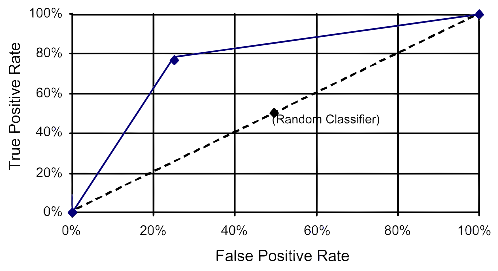

Fig. 3. Example of an ROC plot. Two classifiers are represented: the solid line is a good performing classifier whereas the dashed line represents a random classifier.

3. Iterative instance adjustment for imbalanced domains: IPADE-ID

In this section, we present and describe the proposed approach in depth, denoted as IPADE for Imbalanced Domains (IPADE-ID). IPADE-ID is influenced by the IG algorithm IPADE, having some features in common with it like its iterative way of working or the usage of adaptive evolutionary techniques to optimize the instances generated up to now. Nevertheless, IPADE-ID features several differences from its predecessor: IPADE-ID presents a new initialization of the prototypes procedure, specifically designed for the algorithm used as base learner; IPADE-ID uses a new evaluation measure (AUC instead of accuracy); IPADE-ID modifies the learning process of the positive class as it allows more optimization steps only for this class and positive instances have more influence on the final solution as they are not usually removed from the final solution. Furthermore, the proposal can be used with any learning algorithm as target classifier, having selected the NN and C4.5 algorithms to be used within the method.

From the imbalanced classification point of view, we consider the IPADE-ID a method more similar to data level approaches. Specifically, it is a hybrid resampling method that obtains a new training set (generated set GS, from the IG perspective) with instances from the original training set and new generated instances that are best positioned to cover properly the classes space. IPADE-ID follows an iterative scheme, where it determines the most appropriate number of instances per class and their positioning for a determined classifier, focusing on the positive class.

In particular, IPADE-ID is supported by three main operations:

1. Initialization of the prototypes: As an iterative procedure, IPADE-ID needs an initial set of prototypes that describe the initial state of the algorithm. It is important to select a good initial set because they guide the search towards good solutions and the prototypes that are selected in this stage are maintained or are slightly modified into the final generated training set. Furthermore, the initialization process that provides good prototypes for a specific classifier may not be adequate for another classifier. Therefore, in Section 3.1 we explain the initialization procedures used, focusing on a initialization process based on decision trees that we have designed to use IPADE-ID with C4.5.
2. Optimization of the prototypes: As a IG technique, the prototypes that we currently have in the population can be improved to better represent every class involved in the classification. It is an interesting step that can create new prototypes in a data area where there may not be any prototypes to obtain the decision regions which can improve the final performance of the model. In order to do so, we use differential evolution techniques following the procedure described in Section 3.2.
3. Prototype Selection to extend the Generated Set: IPADE-ID automatically decides how many prototypes are needed to

represent each class, therefore, we need to incorporate a procedure that enables this functionality. The procedure decides if a class needs more prototypes to be properly represented and searches for them in this case. In an imbalanced scenario is important to pay attention to the positive class, therefore, we focus on the selection of positive prototypes easing the selection conditions for this class prototypes. This process is fully described in Section 3.3.

Fig. 4 depicts a flowchart of the IPADE-ID algorithm, outlining its more general operations and delimiting its way of working. The detailed structure of IPADE-ID is shown in Fig. 5, where we can find the pseudocode of the model proposed. We will refer to the number of instructions of the pseudocode shown in the following sections.

3.1. *Initialization of the prototypes based on decision trees*

A random selection (stratified or not) of examples from *TR* may not be the most adequate procedure to initialize the *GS*. Although IPADE-ID iteratively learns instances to find the most appropriate structure of *GS*, a good initialization process can lead the search to better results specially when it is dependent on the target classifier. Instruction 1 generates the initial solution *GS* depending on the classifier. In this step, we have designed two different initialization processes for NN and C4.5 respectively, so that, *GS* covers the entire search space accordingly to its respective classifier.

On the first hand, in NN classification, we focus on the initialization process that was satisfactorily used by the approaches proposed in [19,62] and the original IPADE algorithm [22]. In these works, *GS* initially covers each class with one instance, using its respective centroid to represent each class distribution.

On the other hand, an initialization of *GS* with an excessive small number of instances can lead the C4.5 algorithm to poor performances, damaging the IPADE-ID process. In previous experiments, we observed that the C4.5 algorithm did not generate appropriate rules if the size of *GS* was too small. For this reason, we have designed an initialization process considering the behavior of C4.5. It proceeds as follows:

1. Build a model (a decision tree) with the C4.5 algorithm over the original training data *TR*. This building step does not perform the last pruning phase and does not try to stop the creation of a leaf using a particular minimum number of instances. The aim of this step is to cover the most interesting areas of the original dataset according to the tree.
2. For each leaf, obtain its related instances of *TR*, that is, the instances that were used to build the corresponding leaf. Then, for the selected instances in each leaf compute its centroid instance.
3. Return the centroid instances of the leaves.

Depending on the problem addressed, we cannot be certain that the centroid of the classes or leaves completely cover the region associated to each class, avoiding misclassifications. Thus, instruction 2 applies the first optimization stage using the initial *GS* composed of centroids of classes or leaves. The optimization stage modifies the instances of *GS* using the movement idea in the *D*-dimensional space, adding or subtracting some quantities to the attribute values of the instances. It is important to point out that we normalize all attributes of the dataset to the [0, 1] range.

This initialization procedure enables the differential evolution optimization to search for the best instances representation for imbalanced domains. The intuition behind this operation is to cover each class assigning points that occupy central positions in the data space. In this manner, the samples initially selected cover as many samples of each class as possible and it is the optimization procedure which determines its validity and whether an instance is better than the current selection or not. In this manner, these initial samples are optimized towards the final population where they are accompanied by new prototypes which help to delimit the class regions as they improve the performance of the trial set.

3.2. *Differential evolution optimization for IPADE-ID*

In this section we briefly describe the use of differential evolution in IG techniques, which was proposed in [22], as a position adjusting of instances scheme. In this work, the underlying idea of this process is extended to any learning algorithm.

First of all, it is necessary to explain the solution codification. In this algorithm, each individual in the population encodes a

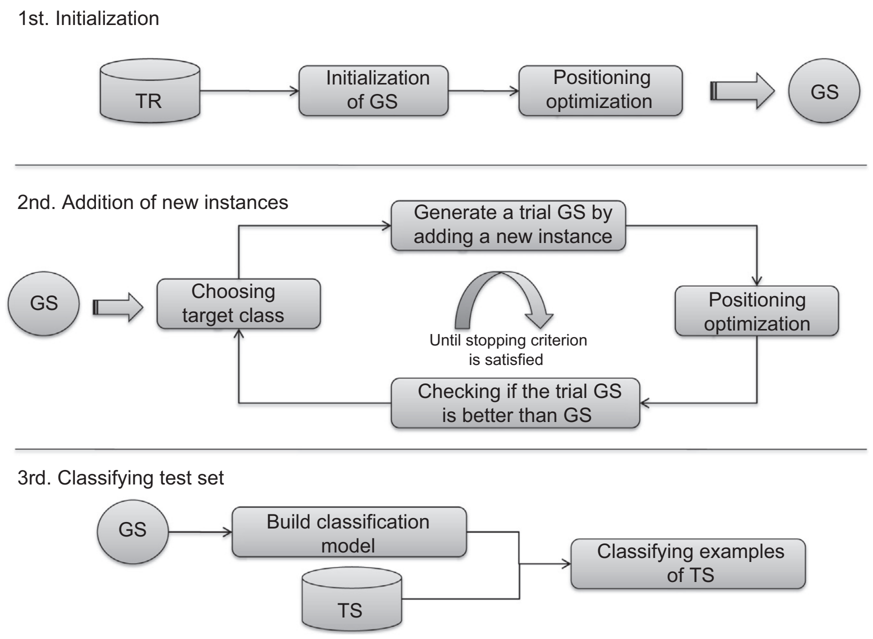
1st. Initialization
TR
Initialization of GS
Positioning optimization
GS
2nd. Addition of new instances
GS
Choosing target class
Generate a trial GS by adding a new instance
Positioning optimization
Until stopping criterion is satisfied
Checking if the trial GS is better than GS
3rd. Classifying test set
GS
Build classification model
TS
Classifying examples of TS

Fig. 4. Flowchart of IPADE-ID.

```
 1: $GS$ = Initialization($TR$, baseClassifier)
 2: DE_Optimization($GS$, $TR$, baseClassifier)
 3: $AUC$ = EvaluateID($GS$, $TR$, baseClassifier)
 4: registerClass[0..$k$] = $optimizable$
 5: numberOfOptimizations[0..$k$] = 0
 6: while $AUC <> 1.0$ or all classes are $non-optimizables$ do
 7:   lessAccuracy = $\infty$
 8:   for $i = 1$ to $k$ do
 9:     if registerClass[$i$] == $optimizable$ then
10:       $AccuracyClass[i]$ = Evaluate ($GS$, Examples of class $i$ in $TR$, baseClassifier)
11:       if $AccuracyClass[i]$ < lessAccuracy then
12:         lessAccuracy = $AccuracyClass[i]$
13:         $targetClass = i$
14:       end if
15:     end if
16:   end for
17:   if $targetClass$==positiveClass and numberOfOptimizations[$targetClass$]>0 then
18:     DE_Optimization($GS_{trial}$, $TR$)
19:   else
20:     $GS_{trial} = GS \cup$ RandomExampleForClass($TR$, $targetClass$)
21:     DE_Optimization($GS_{trial}$, $TR$)
22:   end if
23:   $AUC_{trial}$ = EvaluateID($GS_{trial}$, $TR$, baseClassifier)
24:   if $AUC_{trial} > AUC$ then
25:     $AUC = AUC_{trial}$
26:     $GS = GS_{trial}$
27:     numberOfOptimizations[$targetClass$] = 0
28:   else
29:     if $targetClass$==positiveClass and numberOfOptimizations[$targetClass$]< $OT$ then
30:       numberOfOptimizations[$targetClass$]++:
31:     else
32:       registerClass[$targetClass$] = $non-optimizable$
33:     end if
34:   end if
35: end while
36: Classify (GS, TS)
```

Fig. 5. IPADE-ID algorithm basic structure.

single instance without the class label and, as such, the dimension of the individuals is equal to the number of attributes of the specific problem.

The differential evolution algorithm uses each instance $\mathbf{p}_u$ of *GS*, provided by the IPADE-ID algorithm, as member of the initial population. Next, mutation and crossover operators guide the optimization of the positioning of each $\mathbf{p}_u$ in the *D*-dimensional space. It is important to point out that these operators only produce modifications in the attributes of the instances of *GS*. Hence, the class value remains unchangeable throughout the evolutionary cycle. We will focus on the well-known *DE/CurrentToRand/1* strategy to generate the trial instances $\mathbf{p}'_u$ because it has reported the best behavior. It can be viewed as

$$\mathbf{p}'_u = \mathbf{p}_u + K \cdot (\mathbf{p}_{r_1} - \mathbf{p}_u) + F \cdot (\mathbf{p}_{r_2} - \mathbf{p}_{r_3}) \quad (3)$$

where the scaling factor *F* is a positive control parameter for scaling the different vectors. *K* is a random number from [0, 1]. The examples $\mathbf{p}_{r_1}, \mathbf{p}_{r_2}, \mathbf{p}_{r_3}$ are randomly extracted from *TR* and they belong to the same class as $\mathbf{p}_u$. In the hypothetical case that *TR* does not contain enough instances of the $\mathbf{p}_u$ class, that is, there is not at least three instances of this class in *TR*, we artificially generate the necessary number of new instances $\mathbf{p}_{r_j}$, $1 \le j \le 3$, with the same class label as $\mathbf{p}_u$, using little random perturbations such as $\mathbf{p}_{r_j} = (p_{u1} + rand[-0.1,\ 0.1], p_{u2} + rand[-0.1, 0.1], \ldots,\ p_{um} + rand[-0.1, 0.1], C_u)$.

After applying this operator, we check if the values belong to the interval [0, 1]. If a computed value is greater than 1, we truncate it to 1, and if is lower than 0, we establish it at 0.

After the mutation process over all the instances of *GS*, we obtain a trial solution *GS*′, which is constituted for each $\mathbf{p}'_u$. The selection operator decides which solution *GS*′ or *GS* should survive for the next iteration. This step is dependent on the target classifier. The corresponding fitness value is measured as the AUC (Eq. (2)) obtained with the target classifier using *GS* to build the corresponding learning model to classify the examples in *TR* (using the leave-one-out validation scheme). We try to maximize this value, so the selection operator can be viewed as follows:

$$GS = \begin{cases} GS' & \text{if } AUC(GS') \ge AUC(GS) \\ GS & \text{Otherwise} \end{cases} \quad (4)$$

The success of differential evolution in solving a specific problem crucially depends on choosing the appropriate control parameter values. In order to guarantee a high quality solution, we use the ideas established in [28] to obtain a self adaptive algorithm. This method was established as the best differential evolution technique for IG in [38]. Instruction 3 calculates the AUC of the initial solution, which is computed using *GS* to build a model that classifies the examples of *TR*. Note that a leave-one-out validation scheme is performed, so that the examples in *TR* that also belong to GS are discarded in the AUC computation.

3.3. Prototype selection to extend the generated set

After the first optimization process, IPADE-ID enters in an iterative loop (Instructions 6-36) to determine which classes need more instances to faithfully represent their class distribution, concentrating on the positive class. In order to do this, we focus on local classification accuracy as the mechanism to measure how a class is currently represented. We define two types of classes. A class C_i is said to be *optimizable* if it allows the addition of new instances to improve its local classification accuracy. Initially, all classes are *optimizable*. IPADE-ID allows the positive class to be *optimizable OT* times in order to permit a better representation of its class distribution (generating more instances), even if its local classification accuracy does not improve. Thus, the optimizer has more iterations to obtain an appropriate positioning of the instances.

The local accuracy of C_i is computed by classifying the examples of *TR* whose class is C_i with the instances kept in *GS* (using the respective classifier). The *target class* will be the *optimizable* class with the least local accuracy registered as the class that is more susceptible of being improved. From instructions 7–16, the algorithm identifies the *target class* in each iteration. Initially, all classes start as *optimizable* (Instruction 4) and the number of optimization processes performed without improvement is initialized to 0 (Instruction 5).

In order to reduce the classification error of the *target class*, IPADE-ID extracts a random example of this class from *TR* and adds this to the current *GS* in a new trial set GS_{trial} (Instruction 20). This addition forces the re-positioning of the instances of GS_{trial} using the optimization process again (Instruction 21). To evaluate the goodness of GS_{trial}, instruction 24 computes its corresponding predictive AUC, building a model with GS_{trial} and classifying the examples of *TR* with a leave-one-out validation scheme.

After this process, we have to ensure that the new positioning of instances of GS_{trial}, generated with the optimizer, has reported a successful improvement of the AUC rate with respect to the previous *GS*. If the AUC of the GS_{trial} is lesser than the AUC of *GS*, IPADE-ID does not add this instance to *GS* and then, if the target class corresponds with the positive class (and it has not been optimized *OT* times), its number of iterations without improvement is increased (Instruction 30), else the class is registered as *non-optimizable*. If the addition of the new instance produces an AUC improvement, the current *GS* is updated: $GS = GS_{trial}$.

Note that the addition mechanism is not performed if the target class corresponds to the positive class and in the previous iteration the optimization process did not find a suitable positioning of the instances that improved the AUC of the current *GS* (Instructions 17 and 18).

The stopping criterion is satisfied when the AUC is 1.0 or all the classes are registered as *non-optimizable*. Finally, the adjusted set of instances *GS* is used to classify the *TS* set with the target classifier (Instruction 36).

4. Experimental framework

In this section, we present the set up of the experimental framework used to develop the analysis of our proposal. We will mention the algorithms selected for the comparison together with their configuration parameters, the imbalanced datasets selected and we will introduce the necessity of the usage of statistical tests.

4.1. Algorithms selected for the study and parameters

In order to test the performance of our approach, IPADE-ID, we have selected representative methods that are used to deal with imbalanced datasets to perform the experimental study. Since we have chosen two learning methodologies, the NN rule [24] and the C4.5 decision tree [25], we have also selected other related strategies to each learning methodology in order to test the performance of the proposal in a suitable context. The selected methods are:

- *Resampling techniques*: As resampling techniques used to deal with the imbalanced classification problem, we have selected the SMOTE algorithm [10], the SMOTE+ENN algorithm and the NCL technique.
- *Cost-sensitive algorithms*: As cost-sensitive solutions we have chosen two specific cost-sensitive algorithms instead of a general cost-sensitive framework related to the selected learning methodologies: a NN cost-sensitive approach (NNCS) [63] and C4.5 Cost-Sensitive (C4.5CS) [64]. We have selected these approaches as they make a specific classifier learning algorithm cost-sensitive, including information in their inner way of running. It is interesting to compare the results of these approaches with IPADE-ID as the results of the proposal are dependent on the base classifier used because its behavior is modified according to the selected learner. To relate the behavior of cost-sensitive techniques with the proposal we use an input cost-matrix with the following values for the incorrect classification of instances: $C(+,-) = IR$ and $C(-,+) = 1$.
- *Advanced lazy learning algorithms*: In order to compare the results of the NN rule in imbalanced domains, we have also chosen several advanced lazy learning algorithms that have a competitive performance in balanced datasets: Center Nearest Neighbor Classifier (CENTER-NN) [65], Adaptive KNN Classifier (KNN-ADAPTIVE) [66], and K Symmetrical Nearest Neighbor Classifier (KSNN) [67]. We compare the IPADE-ID-NN version of the proposal with advanced lazy learning techniques using SMOTE as preprocessing algorithm to check the validity of our proposal as it is dependent on the base classifier used.
- *Rule learning approaches for classification*: As related classifiers to the C4.5 algorithm that can be examined in contrast to it, we have selected the Ripper [57] crisp rule learning generator and the Fuzzy Hybrid Genetics-Based Machine Learning (FH-GBML) algorithm [68]. We have chosen the combination of well-known rule learners for classification with SMOTE preprocessing as suitable competitors to the IPADE-ID-C4.5 proposal which is focused on the obtaining of a good decision tree for the imbalanced dataset.

Table 2
Parameter specification for the algorithms tested in the experimentation.

Algorithm	Parameters
SMOTE	$k=5$, distance = Euclidean, balancing = YES
SMOTE+ENN	$kSMOTE=5$, $kENN=3$, distance = Euclidean, balancing = YES
NCL	$k=5$
IPADE-ID	Iterations of Basic DE = 500, iterSFGSS = 8, iterSFHC = 20, Fl = 0.1, Fu = 0.9, $OT=5$
NN, NNCS, KNN-ADAPTIVE, KSNN	$k=1$, distance = Euclidean
C4.5, C4.5CS	Pruned tree, confidence = 0.25, 2 examples per leaf
Ripper	$k=2$, grow set = 0.66, Number of fuzzy rules: $5 \cdot d$ (max. 50 rules), number of rule sets: 200, Crossover probability: 0.9, mutation probability: $1/d$, number of replaced Rules: all rules except the best-one (Pittsburgh-part, elitist approach) Number of rules/5 (GCCL-part), total number of generations 1000, don't Care probability 0.5, probability of the application of the GCCL iteration 0.5
FH-GBML	

The configuration parameters used for these algorithms are shown in Table 2. All the methods were run using KEEL software[3] [69], following the recommended parameter values given in the KEEL platform to configure the methods, which were selected according to the recommendation of the corresponding authors of each algorithm, assuming that the choice of the values of the parameters was optimal.

4.2. Datasets and data partitions

In order to analyze the quality of the proposal against the algorithms introduced in the previous section, we have selected several highly imbalanced datasets. As there is no consensus in the literature about when a dataset is considered highly imbalanced, we will consider that an IR above 9 represents a high IR in a dataset, due to the fact that ignoring the minority class instances by a classifier supposes an error of 0.1 in accuracy, which has poor relevance. Therefore, we have selected 44 datasets from KEEL dataset repository[4] [31] with an IR greater than 9. The data are summarized in Table 3,

[3] http://www.keel.es/.

[4] http://www.keel.es/datasets.php.

Table 3
Summary of highly imbalanced datasets.

Datasets	#Ex.	#Atts.	Class (−; +)	%Class (−; +)	IR
ecoli034vs5	200	7	(p,imL,imU; om)	(10.00, 90.00)	9.00
yeast2vs4	514	8	(cyt; me2)	(9.92, 90.08)	9.08
ecoli067vs35	222	7	(cp,omL,pp; imL,om)	(9.91, 90.09)	9.09
ecoli0234vs5	202	7	(cp,imS,imL,imU; om)	(9.90, 90.10)	9.10
glass015vs2	172	9	(build-win-non_float-proc,tableware, build-win-float-proc; ve-win-float-proc)	(9.88, 90.12)	9.12
yeast0359vs78	506	8	(mit,me1,me3,erl; vac,pox)	(9.88, 90.12)	9.12
yeast02579vs368	1004	8	(mit,cyt,me3,vac,erl; me1,exc,pox)	(9.86, 90.14)	9.14
yeast0256vs3789	1004	8	(mit,cyt,me3,exc; me1,vac,pox,erl)	(9.86, 90.14)	9.14
ecoli046vs5	203	6	(cp,imU,omL; om)	(9.85, 90.15)	9.15
ecoli01vs235	244	7	(cp,im; imS,imL,om)	(9.83, 90.17)	9.17
ecoli0267vs35	224	7	(cp,imS,omL,pp; imL,om)	(9.82, 90.18)	9.18
glass04vs5	92	9	(build-win-float-proc,containers; tableware)	(9.78, 90.22)	9.22
ecoli0346vs5	205	7	(cp,imL,imU,omL; om)	(9.76, 90.24)	9.25
ecoli0347vs56	257	7	(cp,imL,imU,pp; om,omL)	(9.73, 90.27)	9.28
yeast05679vs4	528	8	(me2; mit,me3,exc,vac,erl)	(9.66, 90.34)	9.35
ecoli067vs5	220	6	(cp,omL,pp; om)	(9.09, 90.91)	10.00
vowel0	988	13	(hid; remainder)	(9.01, 90.99)	10.10
glass016vs2	192	9	(ve-win-float-proc; build-win-float-proc, build-win-non_float-proc,headlamps)	(8.89, 91.11)	10.29
glass2	214	9	(Ve-win-float-proc; remainder)	(8.78, 91.22)	10.39
ecoli0147vs2356	336	7	(cp,im,imU,pp; imS,imL,om,omL)	(8.63, 91.37)	10.59
led7digit02456789vs1	443	7	(0,2,4,5,6,7,8,9; 1)	(8.35, 91.65)	10.97
glass06vs5	108	9	(build-win-float-proc,headlamps; tableware)	(8.33, 91.67)	11.00
ecoli01vs5	240	6	(cp,im; om)	(8.33, 91.67)	11.00
glass0146vs2	205	9	(build-win-float-proc,containers,headlamps, build-win-non_float-proc;ve-win-float-proc)	(8.29, 91.71)	11.06
ecoli0147vs56	332	6	(cp,im,imU,pp; om,omL)	(7.53, 92.47)	12.28
cleveland0vs4	177	13	(0; 4)	(7.34, 92.66)	12.62
ecoli0146vs5	280	6	(cp,im,imU,omL; om)	(7.14, 92.86)	13.00
ecoli4	336	7	(om; remainder)	(6.74, 93.26)	13.84
yeast1vs7	459	8	(nuc; vac)	(6.72, 93.28)	13.87
shuttle0vs4	1829	9	(Rad Flow; Bypass)	(6.72, 93.28)	13.87
glass4	214	9	(containers; remainder)	(6.07, 93.93)	15.47
page-blocks13vs2	472	10	(graphic; horiz.line,picture)	(5.93, 94.07)	15.85
abalone9vs18	731	8	(18; 9)	(5.65, 94.25)	16.68
glass016vs5	184	9	(tableware; build-win-float-proc, build-win-non_float-proc,headlamps)	(4.89, 95.11)	19.44
shuttle2vs4	129	9	(Fpv Open; Bypass)	(4.65, 95.35)	20.50
yeast1458vs7	693	8	(vac; nuc,me2,me3,pox)	(4.33, 95.67)	22.10
glass5	214	9	(tableware; remainder)	(4.20, 95.80)	22.81
yeast2vs8	482	8	(pox; cyt)	(4.15, 95.85)	23.10
yeast4	1484	8	(me2; remainder)	(3.43, 96.57)	28.41
yeast1289vs7	947	8	(vac; nuc,cyt,pox,erl)	(3.17, 96.83)	30.56
yeast5	1484	8	(me1; remainder)	(2.96, 97.04)	32.78
ecoli0137vs26	281	7	(pp,imL; cp,im,imU,imS)	(2.49, 97.51)	39.15
yeast6	1484	8	(exc; remainder)	(2.49, 97.51)	39.15
abalone19	4174	8	(19; remainder)	(0.77, 99.23)	128.87

where we denote the number of examples (#Ex.), number of attributes (#Atts.), class name of each class (positive and negative), class attribute distribution and IR. This table is in ascending order according to the IR.

To develop the different experiments we consider a 5-*fold stratified cross-validation model*, for instance, 5 random partitions of data with a 20% maintaining the a priori probabilities of each class and the combination of 4 of them (80%) as training and the remaining ones as test. For each dataset we consider the average results of the five partitions. The datasets used in this study use the partitions provided by the KEEL dataset repository in the imbalanced classification dataset section.[5]

4.3. Statistical tests for performance comparison

In order to support the analysis of the results obtained through an experimentation process, we use hypothesis testing techniques to find significant differences between the studied methods [70]. We consider the use of non-parametric tests, according to the recommendations made in [33,34,70] where a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers is presented. Specifically, we use non-parametric tests due to the fact that the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, causing the statistical analysis to lose credibility [71].

The interesting comparisons that can be made from our experimental study involve the use of the Iman-Davenport test, which is in charge of the detection of statistical differences among a group of results, and the Holm post-hoc test in order to find which algorithms are distinctive among a $1 \times n$ comparison. A post-hoc procedure allows us to know whether a hypothesis of comparison of means could be rejected at a specified level of significance α. Nevertheless, it is quite interesting to compute the p-value associated with each comparison, which represents the lowest level of significance of a hypothesis that results in a rejection. That is the adjusted p-value. In this manner, we can know whether two algorithms are significantly different and how different they are. We also obtain the average ranking of the algorithms, according to the Friedman procedure, which tries to show the performance of an algorithm with respect to the others and is based on the ranking of the algorithms in each dataset.

[5] http://www.keel.es/imbalanced.php.

Table 4
Detailed results table for the algorithms used in imbalanced datasets. Only test results are shown.

Dataset	SMOTE+NN	SMOTE+ENN+NN	NCL+NN	NNCS	SMOTE+ CENTER-NN	SMOTE+ KNN-ADAPTIVE	SMOTE+KSNN	IPADE-ID-NN	SMOTE+C4.5	SMOTE+ ENN+C4.5	NCL+C4.5	C4.5CS	SMOTE+ Ripper	SMOTE+ FH-GBML	IPADE-ID-C4.5
ecoli034vs5	0.8472	0.8472	0.8639	0.8806	0.8417	0.8556	0.8500	**0.9000**	0.9000	0.8806	0.8056	**0.9250**	0.8778	0.8944	0.8833
yeast2vs4	0.8807	0.8828	0.8915	0.8785	0.8089	0.8758	0.8807	**0.9424**	0.8588	0.9042	0.8670	0.8866	0.8703	0.9073	**0.9118**
ecoli067vs35	0.8625	0.8600	**0.8800**	0.8725	0.8700	0.8700	0.8575	0.8350	0.8500	0.8125	0.8400	**0.8825**	0.8450	0.8125	0.8325
ecoli0234vs5	0.8530	0.8780	0.8613	0.8725	0.8337	0.8585	0.8502	**0.9002**	0.8974	0.8947	0.8557	0.8334	0.8643	0.8572	**0.9197**
glass015vs2	0.6573	0.6906	**0.7573**	0.6315	0.7500	0.5992	0.6540	0.6476	0.6772	**0.7957**	0.6599	0.6035	0.6890	0.6008	0.7559
yeast0359vs78	0.7543	**0.7588**	0.7444	0.7413	0.7006	0.7382	0.7576	0.7095	0.7047	0.7024	0.6394	0.6765	0.6784	0.7226	**0.7255**
yeast02579vs368	0.9044	0.9138	0.8946	0.8866	0.8318	**0.9154**	0.9044	0.8966	**0.9143**	0.9138	0.8460	0.8996	0.8812	0.9099	0.8930
yeast0256vs3789	0.7807	0.7860	**0.8059**	0.7645	0.7271	0.7883	0.7840	0.8001	**0.7951**	0.7817	0.7453	0.7846	0.7559	0.7851	0.7810
ecoli046vs5	0.8642	0.8838	0.8669	0.8864	0.8646	0.8669	0.8642	**0.8980**	0.8701	0.8869	0.8364	0.8310	**0.9119**	0.8326	0.9033
ecoli01vs235	0.8286	0.8536	0.8495	0.8755	0.8400	0.8845	0.8264	**0.8982**	0.8377	0.8332	0.7645	0.7641	0.8059	0.8075	**0.8559**
ecoli0267vs35	0.8976	0.8926	0.8326	**0.9325**	0.8326	0.8801	0.8976	0.8252	0.8155	0.8179	0.8102	**0.8527**	0.8478	0.8331	0.8228
glass04vs5	0.9691	0.9629	0.9007	0.9695	0.9574	**0.9754**	0.9691	0.9441	0.9816	0.9754	0.9941	0.9941	0.9757	0.9673	**0.9941**
ecoli0346vs5	0.8838	0.8838	**0.8919**	0.8811	0.8838	0.8811	0.8838	0.8453	0.8980	0.8980	0.8196	0.8507	**0.9264**	0.8331	0.8311
ecoli0347vs56	0.8834	0.9034	0.8963	0.9119	0.8655	0.8941	0.8834	**0.9120**	0.8568	0.8546	0.7536	0.7586	0.8773	0.8600	**0.9253**
yeast05679vs4	0.7753	**0.7969**	0.7835	0.7845	0.6663	0.7590	0.7774	0.7839	0.7602	0.7802	0.7533	0.7243	0.7408	0.8064	**0.8117**
ecoli067vs5	0.8675	0.8300	0.8475	**0.8900**	0.8450	0.8825	0.8675	0.8450	0.8475	0.8450	**0.8850**	0.8825	0.8500	0.8338	0.8775
vowel0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9944	**1.0000**	0.9488	0.9505	0.9455	0.9583	0.9422	0.9578	0.9561	**0.9805**
glass016vs2	0.6814	0.6119	0.6262	0.6474	**0.6893**	0.6183	0.6814	0.6564	0.6062	0.6388	0.6288	0.6155	0.6371	0.6343	**0.7755**
glass2	0.6447	0.6346	0.6653	0.6260	**0.7247**	0.6111	0.6472	0.6578	0.6390	0.7457	0.5557	0.6416	0.6217	0.6771	**0.7665**
ecoli0147vs2356	0.8507	0.8674	0.8707	0.8507	0.8023	0.8771	0.8523	**0.8890**	0.8277	0.8228	0.8385	0.8772	**0.9027**	0.8508	0.8890
led7digit02456789vs1	0.8108	0.8283	0.5143	0.8337	0.7506	0.8502	0.8621	**0.8698**	**0.8908**	0.8379	0.8776	0.8436	0.8502	0.8839	0.8802
glass06vs5	0.9400	0.9900	0.9900	0.9800	0.9300	**0.9950**	0.9300	0.9400	0.9147	0.9647	0.9950	0.9950	0.9545	0.9320	**0.9950**
ecoli01vs5	0.8545	0.8818	0.8636	0.9045	0.8455	0.8614	0.8523	**0.9136**	0.7977	0.8250	0.8386	0.8182	**0.9136**	0.8989	0.8318
glass0146vs2	0.6453	0.6426	0.6428	0.6321	**0.7987**	0.6465	0.6452	0.7019	0.7842	0.7095	0.5959	0.6797	0.6250	0.7064	**0.8500**
ecoli0147vs56	0.8756	0.8923	0.9054	0.9058	0.8988	**0.9070**	0.8756	0.8889	0.8592	0.8424	0.8702	0.8539	0.8559	0.8045	**0.8824**
cleveland0vs4	0.8543	0.8543	0.8325	0.8719	0.8543	0.7937	0.8573	**0.8923**	**0.7878**	0.7512	0.7185	0.6823	0.7090	0.7520	0.7285
ecoli0146vs5	0.8481	0.8731	0.8596	0.8808	0.8481	0.8615	0.8481	**0.8865**	0.8981	0.8981	0.7865	0.8385	0.8615	**0.9202**	0.9135
ecoli4	0.9171	0.9123	0.8623	**0.9357**	0.8778	0.9187	0.9171	0.9171	0.7794	0.9044	0.8687	0.8636	0.8842	**0.9302**	0.8842
yeast1vs7	0.7479	0.7255	0.6902	0.7145	0.7129	0.7150	**0.7514**	0.7206	0.7003	0.7371	0.6015	0.6139	**0.9997**	0.7191	0.6529
shuttle0vs4	0.9960	0.9960	0.9960	**1.0000**	0.9957	**1.0000**	0.9960	0.9960	0.9997	0.9997	0.9997	0.9997	0.6583	0.9980	**1.0000**
glass4	0.8917	**0.9101**	0.8892	0.8818	0.8442	0.8492	0.8917	0.8918	0.8867	0.8650	0.8700	0.8431	**0.8967**	0.8867	0.8300
page-blocks13vs4	0.9977	0.9777	0.9800	**1.0000**	0.9989	0.9800	0.9955	0.9620	0.9955	0.9910	**0.9955**	0.9789	0.9888	0.9515	0.9788
abalone9-18	0.6820	0.6947	0.6916	0.7236	0.6859	0.6515	0.6827	**0.7804**	0.6283	**0.7193**	0.7021	0.6126	0.5303	0.7165	0.6859
glass016vs5	0.8771	0.8800	**0.9300**	0.9157	0.9186	0.8886	0.8743	0.8943	0.8129	0.8629	0.9857	**0.9886**	0.9486	0.8993	0.8943
shuttle2vs4	**1.0000**	**1.0000**	0.9500	0.9920	**1.0000**	**1.0000**	0.9960	0.9460	0.9917	**1.0000**	**1.0000**	**1.0000**	0.9958	0.9940	**1.0000**
yeast1458vs7	0.6390	**0.6912**	0.6305	0.6702	0.6104	0.5766	0.6443	0.6357	0.5367	0.5563	0.4925	0.5540	0.6315	0.6287	**0.6344**
glass5	0.8829	0.8707	**0.9280**	0.9256	0.8829	0.7902	0.8829	0.8854	0.8805	0.7756	0.9378	0.9427	0.9329	0.7671	**0.9976**
yeast2vs8	0.8055	0.7969	**0.8131**	0.7915	0.7577	0.7935	0.7816	0.7783	0.8338	0.8197	0.6250	**0.8652**	0.8457	0.7442	0.7002
yeast4	0.7242	0.7607	0.7552	0.7923	0.7321	0.6887	0.7259	**0.8073**	0.7121	0.7257	0.6967	0.7222	0.7642	0.8137	**0.8167**
yeast1289vs7	0.6444	0.6594	0.5722	0.7008	0.6056	0.5831	0.6460	**0.7382**	0.6832	0.6332	0.5273	0.6769	**0.7365**	0.7238	0.6841
yeast5	0.9326	**0.9503**	0.9347	0.9344	0.8698	0.8931	0.9326	0.9476	0.9337	0.9406	0.9035	0.9330	0.9323	**0.9469**	0.9424
ecoli0137vs26	0.8281	0.8281	0.8391	0.8208	0.8263	0.8409	0.8281	**0.8727**	0.8136	0.8136	**0.8481**	0.8281	0.7922	0.8236	0.7208
yeast6	0.7998	0.8351	0.8161	0.8472	0.8084	0.7955	0.8001	**0.8562**	0.8294	0.8270	0.7785	0.8082	0.8208	**0.8646**	0.8384
abalone19	0.5176	0.5135	0.5063	0.5900	0.5191	0.4986	0.5183	**0.6574**	0.5205	0.5166	0.5000	0.5701	**0.7063**	0.6708	0.5519
Mean	0.8272	0.8342	0.8210	0.8416	0.8161	0.8183	0.8278	**0.8435**	0.8172	0.8238	0.7925	0.8122	0.8262	0.8263	**0.8416**

Table 5
Average Friedman rankings and adjusted *p*-values using Holm's post-hoc procedure for the NN versions compared in the study.

Algorithm	Average Friedman ranking	Adjusted *p*-value
IPADE-ID-NN	3.5682	
NNCS	3.6364	0.8961
SMOTE+ENN+NN	4.0341	0.7446
NCL+NN	4.3295	0.4346
SMOTE+KSNN	4.8750	**0.0577**
SMOTE+KNN-ADAPTIVE	4.8977	**0.0577**
SMOTE+NN	4.9205	**0.0577**
SMOTE+CENTER-NN	5.7386	**0.0002**

Table 6
Average runtime (seconds) of versions compared in this study.

Algorithm	Average runtime
IPADE-ID-NN	11.31
NNCS	21.39
SMOTE+ENN+NN	5.84
NCL+NN	6.59
SMOTE+KSNN	14.27
SMOTE+KNN-ADAPTIVE	14.32
SMOTE+NN	4.55
SMOTE+CENTER-NN	18.07

These tests are suggested in the studies presented in [33,34,70], where their use in the field of machine learning is recommended. For a wider description of the use of these tests, refer to the website on *Statistical Inference in Computational Intelligence and Data Mining*.[6]

5. Experimental results and analysis

In this section, we present the empirical analysis of the proposed IPADE-ID algorithm in order to determine its robustness in a scenario of highly imbalanced datasets. We divide the study in several parts: a first one devoted to the results of IPADE-ID using the NN rule in its way of working (Section 5.1), and a second part with the results of the proposal using the C4.5 decision tree as classifier (Section 5.2). Finally, a study on the impact of the data modification that some of the algorithms perform is included in Section 5.3.

5.1. Analysis of IPADE-ID using the NN rule

The following part of the study will consider the performance of the IPADE-ID algorithm using the NN rule as learning method, in contrast with several solutions provided to deal with imbalanced datasets and other lazy learning approaches. Table 4 shows the average AUC results in test for each of the methods tested in every selected dataset, namely, the basic NN rule using SMOTE, SMOTE+ENN and NCL; the NN cost-sensitive version; Center-NN; KNN-Adaptive; KSNN; and the proposed IPADE-ID-NN. Note that the advanced lazy learning algorithms (CENTER-NN, KNN-ADAPTIVE and KSNN) use SMOTE preprocessing not to be in disadvantage due to the imbalance. The best result per dataset is highlighted in bold.

Taking a quick glance at this table shows that the best performing method in average over the highly imbalanced datasets is the proposal IPADE-ID-NN. However, we need to check if this supposition can be supported by non-parametric statistical tests. We first use an Iman-Davenport test to check if there are differences between the performance of the algorithms.

Table 7
Average Friedman rankings and adjusted *p*-values using Holm's post-hoc procedure for the rule learning versions compared in the study.

Algorithm	Average Friedman ranking	Adjusted *p*-value
IPADE-ID-C4.5	2.8750	
SMOTE+FH-GBML	3.7159	**0.0679**
SMOTE+Ripper	3.8636	**0.0637**
SMOTE+ENN+C4.5	4.0227	**0.0381**
SMOTE+C4.5	4.0909	**0.0332**
C4.5CS	4.4432	**0.0033**
NCL+C4.5	4.9886	**0.0000**

The *p*-value (0.0001) that is computed is low enough to reject the null equality hypothesis with a high confidence level. Therefore, as we know there are significant differences, we proceed with the application of the post-hoc procedure. In Table 5 we can see the average ranks of the algorithms and the adjusted *p*-values that were calculated using Holm's post-hoc procedure. As we expected, IPADE-ID-NN obtains the lower ranking of the algorithms used which turns our proposal into the control method. The adjusted *p*-values associated to all the methods that have been preprocessed using the SMOTE algorithm are low enough to reject the null-hypothesis with a high confidence level (highlighted in bold). The other approaches, where the null-hypothesis is not rejected, introduce NN components in the previous sampling step which indirectly increases the final performance when using NN as basic classifier.

Table 6 shows the average runtime[7] of the comparison algorithms over all the considered datasets. Despite of its evolutionary nature, we can observe that IPADE-ID is very competitive in terms of efficacy when the NN rule is used as target classifier.

5.2. Analysis of IPADE-ID using the C4.5 decision tree

In this second step of the study, we will compare the performance of the IPADE-ID algorithm using the C4.5 decision tree as learning method with well-known solutions used to solve the imbalanced classification problem and other rule learning solutions. In Table 4 the average AUC results in test for each of the methods tested are shown. Specifically the methods presented are the basic C4.5 decision tree used in combination with SMOTE, SMOTE+ENN and NCL; C4.5 Cost-Sensitive; Ripper using SMOTE preprocessing; FH-GBML with datasets preprocessed with SMOTE; and the proposed IPADE-ID-C4.5. The best result per dataset is highlighted in bold, as in the previous section.

The results table reveals a similar situation as in the previous case: the proposal IPADE-ID-C4.5 is the method that obtains the best average AUC value over all the tested datasets. Once again, we need to validate the performance of our proposal using statistical tests. We start computing the *p*-value (0.0004) associated to the Iman-Davenport test, which is low enough to reject the null-hypothesis with a high confidence level. Consequently, we apply Holm's post-hoc procedure in order to categorize the differences between algorithms. Table 7 shows the average Friedman ranks of the algorithms and the adjusted *p*-values computed by the aforementioned post-hoc procedure. Following our intuitions, IPADE-ID-C4.5 obtains the best ranking among the algorithms becoming the control method. As all the adjusted *p*-values are sufficiently low to reject the null-hypothesis in all cases, the assumption where IPADE-ID-C4.5 is the best performing method considered for highly imbalanced datasets is reinforced.

[6] http://sci2s.ugr.es/sicidm/.

[7] These results have been obtained with an Intel(R) Core(TM) i7 CPU 920 at 2.67 GHz.

Table 8 presents the average runtime of IPADE-ID-C4.5 over all the considered datasets in comparison with the rest of rule learning algorithms. In this case, IPADE-ID is slower than most of the comparison algorithms. As we stated in Section 3.1, C4.5 needs more prototypes in *GS* to generate appropriate rules, and therefore, the optimization process takes more time. Nevertheless, this additional computation time is admissible considering the AUC achieved.

Table 8
Average Runtime (seconds) of versions compared in this study.

Algorithm	Average runtime
IPADE-ID-C4.5	386.57
SMOTE+FH-GBML	11 809.61
SMOTE+Ripper	24.07
SMOTE+ENN+C4.5	19.86
SMOTE+C4.5	18.61
C4.5CS	7.80
NCL+C4.5	9.48

5.3. *Analysis of the data used at the classification step*

In this part of the study, our aim is to show how the original data used to classify is modified by certain algorithms. This information is very interesting in an imbalanced scenario as standard classifiers behavior is not heavily deteriorated when the data used to build the classification model is somehow balanced. We aim to understand how some of the algorithms used in the study alter the original data and see if the data distribution has an impact on the final performance of the classifiers. Table 9 shows the resulting IR, in each dataset, of all the techniques that modify the original *TR*. For IPADE-ID, the obtained *GS* is analyzed, comparing the number of instances belonging to the majority and minority classes.

Observing this table, we can observe the following conducts:

- SMOTE produces balanced datasets as expected with the parameter setup used. SMOTE+ENN tends to remove more positive instances than negative ones (in proportion), therefore, its IR is slightly higher than the obtained with SMOTE.
- Since NCL is based on a data reduction process of the majority class, its IR is very similar to the original data.

Table 9
IR of the techniques that modify the *TR*.

Dataset	Original data	SMOTE	SMOTE-ENN	NCL	IPADE-ID-NN	IPADE-ID-C45
ecoli034vs5	9.0000	1.0000	1.0056	8.1250	1.1667	3.3571
yeast2vs4	9.0800	1.0000	1.0291	7.8775	4.2000	8.4800
ecoli067vs35	9.0900	1.0000	1.0260	7.7159	0.3333	0.8889
ecoli0234vs5	9.1000	1.0000	1.0126	8.1500	0.3750	1.7027
glass015vs2	9.1200	1.0000	1.0671	6.3235	2.0000	0.6667
yeast0359vs78	9.1200	1.0000	1.0276	6.7500	0.3333	1.0000
yeast02579vs368	9.1400	1.0000	1.0054	8.2626	1.2143	1.6250
yeast0256vs3789	9.1400	1.0000	1.0276	7.4975	1.4545	2.3061
ecoli046vs5	9.1500	1.0000	1.0282	8.2000	2.0833	0.7692
ecoli01vs235	9.1700	1.0000	1.0128	7.8125	0.6667	1.3846
ecoli0267vs35	9.1800	1.0000	1.0271	7.7500	0.8667	1.6552
glass04vs5	9.2200	1.0000	1.0219	7.8333	1.0000	1.5424
ecoli0346vs5	9.2500	1.0000	1.0027	8.3625	0.8182	1.4800
ecoli0347vs56	9.2800	1.0000	1.0088	8.3300	2.5455	2.3913
yeast05679vs4	9.3500	1.0000	1.0465	7.5196	0.4118	1.5000
ecoli067vs5	10.0000	1.0000	1.0568	8.6375	0.2632	3.1000
vowel0	10.1000	1.0000	1.0003	9.8500	0.5556	1.1458
glass016vs2	10.2900	1.0000	1.0690	7.3676	0.9231	1.4074
glass2	10.3900	1.0000	1.1399	8.4265	1.0000	44.8000
ecoli0147vs2356	10.5900	1.0000	1.0133	9.4914	0.5000	2.0000
led7digit02456789vs1	10.9700	1.0000	1.2462	9.7973	2.4286	1.2791
glass06vs5	11.0000	1.0000	1.0156	9.3333	2.7143	3.8125
ecoli01vs5	11.0000	1.0000	1.0307	9.9875	1.0000	1.5833
glass0146vs2	11.0600	1.0000	1.0638	7.8971	1.0000	1.9200
ecoli0147vs56	12.2800	1.0000	1.0554	11.4200	5.4000	2.6667
cleveland0vs4	12.6200	1.0000	1.0064	10.6538	3.6250	1.3030
ecoli0146vs5	13.0000	1.0000	1.0340	11.9750	0.8333	1.1852
ecoli4	13.8400	1.0000	1.0212	14.9375	0.8462	1.3714
yeast1vs7	13.8700	1.0000	1.0418	11.4917	3.1000	7.3333
shuttle0vs4	13.8700	1.0000	1.0006	13.8293	1.5000	1.6842
glass4	15.4700	1.0000	1.0710	13.5769	1.8889	2.3333
page-blocks13vs4	15.8500	1.0000	1.0045	15.5179	1.0000	1.3103
abalone9-18	16.6800	1.0000	1.1058	13.1071	0.3571	0.6970
glass016vs5	19.4400	1.0000	1.0073	17.6944	0.7692	1.1250
shuttle2vs4	20.5000	1.0000	1.0293	18.6250	1.4706	1.0702
yeast1458vs7	22.1000	1.0000	1.0355	18.4667	0.3846	1.5263
glass5	22.8100	1.0000	1.0474	20.7500	0.5000	1.2667
yeast2vs8	23.1000	1.0000	1.0549	21.1375	1.5000	2.3043
yeast4	28.4100	1.0000	1.0695	25.5490	0.4167	0.6875
yeast1289vs7	30.5600	1.0000	1.0256	26.8083	1.4615	3.0417
yeast5	32.7800	1.0000	1.0253	31.6477	2.3000	1.6591
ecoli0137vs26	39.1500	1.0000	1.0037	37.2857	1.2500	5.2500
yeast6	39.1500	1.0000	1.0464	39.4571	0.4167	1.3182
abalone19	128.8700	1.0000	1.0353	124.7891	1.7778	2.8780
Mean	17.4350	1.0000	1.0388	15.8186	1.3784	3.0411

- In general, the IPADE-ID scheme does not need to balance classes to obtain good generalization results. The generated dataset is very dependent of the problem tackled and the used classifier, so that the corresponding IR is not related to the original IR. It is noteworthy that in many problems, the IPADE-ID algorithm generates more positives instances with respect to the negative ones because it is the most complex class.
- The main difference between IPADE-ID-NN and IPADE-ID-C4.5 is that the NN version is more balanced. It means that C4.5 needs more instances in the negative class to represent its decision boundaries.

To sum up, our experimental study has shown that IPADE-ID is an algorithm that presents a good performance in the scenario of highly imbalanced datasets. The integration of instance generation techniques in a resampling step produces an improvement of the results in this unfavorable scenario. Furthermore, the method has shown its robustness with two different learning paradigms, taking into consideration the features of the learning methods into the way of working of the proposal.

6. Concluding remarks

In this paper, we have presented IPADE-ID, a new approach to deal with the problem of classification with highly imbalanced datasets. The proposal provides a solution that modifies the training set using a IG technique based on differential evolution as base for the procedure, adapting its way of working to this imbalanced scenario. As learning methods, we have selected the NN rule and the C4.5 decision tree and we have adapted the IPADE-ID approach according to these methods behavior.

The experimental study performed has shown that the usage of instance generation techniques to deal with highly imbalanced datasets can be taken into consideration as a valid solution to this problem. IPADE-ID has demonstrated its good behavior in an exhaustive comparison with methodologies that are used to solve this problem such as resampling techniques or cost-sensitive solutions. The proposal outperforms the other approaches in the scenario of highly imbalanced datasets, which usually is a scenario where most algorithms have lots of difficulties to perform properly. On the other hand, the proposal obtains a good performance using different techniques as learning methods, which makes it extensible to other learning paradigms and permits a further adaptation of the presented proposal into more powerful solutions that adapt the procedure at the algorithm level.

Acknowledgments

This work was partially supported by the Spanish Ministry of Science and Technology under project TIN2011-28488 and the Andalusian Research Plans P11-TIC-7765 and P10-TIC-6858. V. López holds a FPU scholarship from Spanish Ministry of Education.

References

[1] 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.
[2] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1263–1284.
[3] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the 17th IEEE International Joint Conference on Artificial Intelligence (IJCAI'01), 2001, pp. 973–978.
[4] B. Zadrozny, J. Langford, N. Abe, Cost-sensitive learning by cost-proportionate example weighting, in: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03), 2003, pp. 435–442.
[5] G.M. Weiss, Mining with rarity: a unifying framework, SIGKDD Explorations 6 (1) (2004) 7–19.
[6] N. Japkowicz, S. Stephen, The class imbalance problem: a systematic study, Intelligent Data Analysis Journal 6 (5) (2002) 429–450.
[7] V. López, A. Fernández, J.G. Moreno-Torres, F. Herrera, Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics, Expert Systems with Applications 39 (7) (2012) 6585–6608.
[8] T. Yu, S. Simoff, T. Jan, VQSVM: a case study for incorporating prior domain knowledge into inductive machine learning, Neurocomputing 73 (13–15) (2010) 2614–2623.
[9] S.-H. Oh, Error back-propagation algorithm for classification of imbalanced data, Neurocomputing 74 (6) (2011) 1058–1061.
[10] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, Journal of Artificial Intelligent Research 16 (2002) 321–357.
[11] G.E.A.P.A. Batista, R.C. Prati, M.C. Monard, A study of the behaviour of several methods for balancing machine learning training data, SIGKDD Explorations 6 (1) (2004) 20–29.
[12] S. García, J. Derrac, I. Triguero, C.J. Carmona, F. Herrera, Evolutionary-based selection of generalized instances for imbalanced classification, Knowledge-Based Systems 25 (1) (2012) 3–12.
[13] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning 38 (3) (2000) 257–286.
[14] I. Kononenko, M. Kukar, Machine Learning and Data Mining: Introduction to Principles and Algorithms, Horwood Publishing Limited, 2007.
[15] S. García, A. Fernández, F. Herrera, Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems, Applied Soft Computing 9 (2009) 1304–1314.
[16] A. de Haro-Garcia, N. Garcia-Pedrajas, A scalable method for instance selection for class-imbalance datasets, in: Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA'11), 2011, pp. 1383–1390.
[17] J. Derrac, S. García, F. Herrera, IFS-CoCo: instance and feature selection based on cooperative coevolution with nearest neighbor rule, Pattern Recognition 43 (6) (2010) 2082–2105.
[18] S. García, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 417–435.
[19] H.A. Fayed, S.R. Hashem, A.F. Atiya, Self-generating prototypes for pattern classification, Pattern Recognition 40 (5) (2007) 1498–1509.
[20] I. Triguero, J. Derrac, S. García, F. Herrera, A taxonomy and experimental study on prototype generation for nearest neighbor classification, IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews 42 (1) (2012) 86–100.
[21] S. García, F. Herrera, Evolutionary under-sampling for classification with imbalanced data sets: proposals and taxonomy, Evolutionary Computation 17 (3) (2009) 275–306.
[22] I. Triguero, S. García, F. Herrera, IPADE: iterative prototype adjustment for nearest neighbor classification, IEEE Transactions on Neural Networks 21 (12) (2010) 1984–1990.
[23] I. Triguero, S. García, F. Herrera, Enhancing IPADE algorithm with a different individual codification, in: Proceedings of the Sixth International Conference on Hybrid Artificial Intelligence Systems (HAIS'11), 2011, pp. 262–270.
[24] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.
[25] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1993.
[26] R. Storn, K.V. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (10) (1997) 341–359.
[27] K.V. Price, R.M. Storn, J.A. Lampinen, Differential evolution: a practical approach to global optimization, Natural Computing Series (2005).
[28] F. Neri, V. Tirronen, Scale factor local search in differential evolution, Memetic Computing 1 (2) (2009) 153–171.
[29] E. Corchado, A. Abraham, A. Carvalho, Hybrid intelligent algorithms and applications, Information Sciences 180 (14) (2010) 2633–2634.
[30] E. Corchado, M. Graña, M. Wozniak, New trends and applications on hybrid artificial intelligence systems, Neurocomputing 75 (1) (2012) 61–63.
[31] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic and Soft Computing 17 (2–3) (2011) 255–287.
[32] 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.
[33] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[34] S. García, F. Herrera, An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.
[35] L. Nanni, A. Lumini, Particle swarm optimization for prototype reduction, Neurocomputing 72 (4–6) (2008) 1092–1097.
[36] J.S. Sánchez, R. Barandela, A.I. Marqués, R. Alejo, J. Badenas, Analysis of new techniques to obtain quality training sets, Pattern Recognition Letters 24 (7) (2003) 1015–1022.
[37] J.S. Sánchez, High training set size reduction by space partitioning and prototype abstraction, Pattern Recognition 37 (7) (2004) 1561–1564.

[38] I. Triguero, S. García, F. Herrera, Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification, Pattern Recognition 44 (4) (2011) 901–916.
[39] T. Kohonen, The self organizing map, Proceedings of the IEEE 78 (9) (1990) 1464–1480.
[40] W.-J. Lin, J. Chen, Biomarker classifiers for identifying susceptible subpopulations for treatment decisions, Pharmacogenomics 13 (2) (2012) 147–157.
[41] I. Brown, C. Mues, An experimental comparison of classification algorithms for imbalanced credit scoring data sets, Expert Systems with Applications 39 (3) (2012) 3446–3453.
[42] J. Xiao, L. Xie, C. He, X. Jiang, Dynamic classifier ensemble model for customer classification with imbalanced class distribution, Expert Systems with Applications 39 (3) (2012) 3668–3675.
[43] W. Khreich, E. Granger, A. Miri, R. Sabourin, Iterative boolean combination of classifiers in the ROC space: an application to anomaly detection with HMMs, Pattern Recognition 43 (8) (2010) 2732–2752.
[44] N. García-Pedrajas, J. Pérez-Rodríguez, M. García-Pedrajas, D. Ortiz-Boyer, C. Fyfe, Class imbalance methods for translation initiation site recognition in dna sequences, Knowledge-Based Systems 25 (1) (2012) 22–34.
[45] A. Orriols-Puig, E. Bernadó-Mansilla, Evolutionary rule-based systems for imbalanced datasets, Soft Computing 13 (3) (2009) 213–225.
[46] V. García, R. Mollineda, J.S. Sánchez, On the k-NN performance in a challenging scenario of imbalance and overlapping, Pattern Analysis Applications 11 (3–4) (2008) 269–280.
[47] G.M. Weiss, F. Provost, Learning when training data are costly: the effect of class distribution on tree induction, Journal of Artificial Intelligence Research 19 (2003) 315–354.
[48] R.C. Prati, G.E.A.P.A. Batista, M.C. Monard, Learning with class skews and small disjuncts, in: Seventeenth Brazilian Symposium on Artificial Intelligence (SBIA2004), 2004, pp. 296–306.
[49] J.G. Moreno-Torres, T. Raeder, R. Alaíz-Rodríguez, N.V. Chawla, F. Herrera, A unifying view on dataset shift in classification, Pattern Recognition 45 (1) (2012) 521–530.
[50] K. Napierala, J. Stefanowski, S. Wilk, Learning from imbalanced data in presence of noisy and borderline examples, in: Seventh International Conference on Rough Sets and Current Trends in Computing (RSCTC2010), 2010, pp. 158–167.
[51] J. Laurikkala, Improving identification of difficult small classes by balancing class distribution, in: Proceedings of the Eighth Conference on AI in Medicine in Europe (AIME'01), 2001, pp. 63–66.
[52] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on System, Man and Cybernetics 2 (3) (1972) 408–421.
[53] H. Han, W. Wang, B. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: Proceedings of the 2005 International Conference on Intelligent Computing (ICIC'05), Lecture Notes in Computer Science, vol. 3644, 2005, pp. 878–887.
[54] H. He, Y. Bai, E. Garcia, S. Li, ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: Proceedings of the 2008 IEEE International Joint Conference Neural Networks (IJCNN'08), 2008, pp. 1322–1328.
[55] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-SMOTE: safe-level-synthetic minority over-sampling Technique for handling the class imbalanced problem, in: Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining PAKDD'09, 2009, pp. 475–482.
[56] T.M. Khoshgoftaar, N. Seliya, D.J. Drown, Evolutionary data analysis for the class imbalance problem, Intelligent Data Analysis 14 (1) (2010) 69–88.
[57] W. Cohen, Fast effective rule induction, in: Proceedings of the 12th International Conference on Machine Learning (ICML'95), 1995, pp. 1–10.
[58] J. Luengo, A. Fernández, S. García, F. Herrera, Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling, Soft Computing 15 (10) (2011) 1909–1936.
[59] E. Frank, I. Witten, Generating accurate rule sets without global optimization, in: Proceedings of the Fifteenth International Conference on Machine Learning, 1998, pp. 144–151.
[60] S. Salzberg, A nearest hyperrectangle learning method, Machine Learning 6 (1991) 251–276.
[61] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (7) (1997) 1145–1159.
[62] M. Lozano, J.M. Sotoca, J.S. Sánchez, F. Pla, E. Pekalska, R.P.W. Duin, Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces, Pattern Recognition 39 (10) (2006) 1827–1838.
[63] R. Barandela, J.S. Sánchez, V. García, E. Rangel, Strategies for learning in class imbalance problems, Pattern Recognition 36 (3) (2003) 849–851.
[64] K.M. Ting, An instance-weighting method to induce cost-sensitive trees, IEEE Transactions on Knowledge and Data Engineering 14 (3) (2002) 659–665.
[65] Q. Gao, Z. Wang, Center-based nearest neighbor classifier, Pattern Recognition 40 (2007) 346–349.
[66] J. Wang, P. Neskovic, L. Cooper, Improving nearest neighbor rule with a simple adaptative distance measure, Pattern Recognition Letters 28 (2007) 207–213.
[67] R. Nock, M. Sebban, D. Bernard, A simple locally adaptive nearest neighbor rule with application to pollution forecasting, International Journal of Pattern Recognition and Artificial Intelligence 17 (2003) 1369–1382.
[68] H. Ishibuchi, T. Yamamoto, T. Nakashima, Hybridization of fuzzy GBML approaches for pattern classification problems, IEEE Transactions on Systems and Man and Cybernetics-Part B: Cybernetics 35 (2) (2005) 359–365.
[69] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing 13 (3) (2009) 307–318.
[70] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Computing 13 (10) (2009) 959–977.
[71] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 5th ed., Chapman & Hall/CRC, 2011.

Victoria López received her M.Sc. degree in Computer Science from the University of Granada, Granada, Spain, in 2009. She is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. Her research interests include data mining, classification in imbalanced domains, fuzzy rule learning and evolutionary algorithms.

Isaac Triguero received the M.Sc. degree in Computer Science from the University of Granada, Granada, Spain, in 2009. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. His research interests include data mining, semisupervised learning, data reduction and evolutionary algorithms.

Cristóbal José Carmona received the M.Sc. and Ph.D. degrees in computer science from the University of Jaén, Spain, in 2006 and 2011, respectively. He is a researcher in the Department of Computer Science, University of Jaén, Spain. Currently, he is working with Intelligent Systems and Data Mining Research Group of Jaén. His research interest includes supervised descriptive rule discovery, subgroup discovery, contrast set mining, emerging pattern mining, evolutionary fuzzy systems, evolutionary algorithm and data mining.

Salvador García received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively. He is currently an Associate Professor in the Department of Computer Science, University of Jaén, Jaén, Spain. He has had more than 25 papers published in international journals. He has co-edited two special issues of international journals on different Data Mining topics. His research interests include data mining, data reduction, data complexity, imbalanced learning, semi-supervised learning, statistical inference and evolutionary algorithms.

Francisco Herrera received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain.

He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has had more than 200 papers published in international journals. He is coauthor of the book "*Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*" (World Scientific, 2001).

He currently acts as Editor in Chief of the international journal "Progress in Artificial Intelligence (Springer) and serves as area editor of the Journal Soft Computing (area of evolutionary and bioinspired algorithms) and International Journal of Computational Intelligence Systems (area of information systems). He acts as associated editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Advances in Fuzzy Systems, and International Journal of Applied Metaheuristics Computing; and he serves as member of several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Knowledge and Information Systems, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, Swarm and Evolutionary Computation.

He received the following honors and awards: ECCAI Fellow 2009, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", and International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010).

His current research interests include computing with words and decision making, data mining, bibliometrics, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.