OligoIS: Scalable Instance Selection for Class-Imbalanced Data Sets

Nicolás García-Pedrajas, Javier Pérez-Rodríguez, and Aida de Haro-García

***Abstract*—In current research, an enormous amount of information is constantly being produced, which poses a challenge for data mining algorithms. Many of the problems in extremely active research areas, such as bioinformatics, security and intrusion detection, or text mining, share the following two features: large data sets and class-imbalanced distribution of samples. Although many methods have been proposed for dealing with class-imbalanced data sets, most of these methods are not scalable to the very large data sets common to those research fields. In this paper, we propose a new approach to dealing with the class-imbalance problem that is scalable to data sets with many millions of instances and hundreds of features. This proposal is based on the divide-and-conquer principle combined with application of the selection process to balanced subsets of the whole data set. This divide-and-conquer principle allows the execution of the algorithm in linear time. Furthermore, the proposed method is easy to implement using a parallel environment and can work without loading the whole data set into memory. Using 40 class-imbalanced medium-sized data sets, we will demonstrate our method's ability to improve the results of state-of-the-art instance selection methods for class-imbalanced data sets. Using three very large data sets, we will show the scalability of our proposal to millions of instances and hundreds of features.**

***Index Terms*—Class-imbalance problem, instance selection, instance-based learning, very large problems.**

I. Introduction

THE overwhelming amount of data that is currently available in any field of research poses new problems for data mining and knowledge discovery methods. This large amount of data makes most existing algorithms inapplicable to many real-world problems. Furthermore, one of the distinctive features of many common problems in data mining applications is the uneven distribution of the instances of the different classes. In extremely active research areas, such as artificial intelligence in medicine, bioinformatics, or intrusion detection, two classes are usually involved: a class of interest or a positive class, and a negative class that is overrepresented in the data sets. This is usually referred to as the class-imbalance problem [1]. In highly imbalanced problems, the ratio between the positive and negative classes can be as high as $1:1000$ or $1:10\,000$.

Manuscript received October 13, 2011; revised April 9, 2012; accepted June 12, 2012. Date of publication July 30, 2012; date of current version January 11, 2013. This work was supported in part by the Spanish Ministry of Science and Innovation under Project TIN2008-03151 and in part by the Junta de Andalucía under Project P09-TIC-4623. This paper was recommended by Associate Editor N. Chawla.

The authors are with the Department of Computing and Numerical Analysis, University of Córdoba, 14071 Córdoba, Spain (e-mail: npedrajas@uco.es; javier.perez@uco.es; adeharo@uco.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2012.2206381

It has been repeatedly shown that most classification methods suffer from an imbalanced distribution of training instances among classes [2]. Most learning algorithms expect an approximately even distribution of instances among the different classes and suffer, to different degrees, when that is not the case. Dealing with the class-imbalance problem is a difficult but relevant task as many of the most interesting and challenging real-world problems have a very uneven class distribution.

Many algorithms and methods have been proposed to ameliorate the effect of class imbalance on the performance of learning algorithms. There are three main approaches to these methods.

- *Internal approaches acting on the algorithm*. These approaches modify the learning algorithm to deal with the imbalance problem. They can adapt the decision threshold to create a bias toward the minority class or introduce costs in the learning process to compensate the minority class.
- *External approaches acting on the data*. These algorithms act on the data instead of the learning method. They have the advantage of being independent from the classifier used. There are two basic approaches: oversampling the minority class and undersampling the majority class.
- *Combined approaches that are based on boosting accounting for the imbalance in the training set*. These methods modify the basic boosting method to account for minority class underrepresentation in the data set.

There are two principal advantages of choosing sampling over cost-sensitive methods. First, sampling is more general as it does not depend on the possibility of adapting a certain algorithm to work with classification costs. Second, the learning algorithm is not modified, which can cause difficulties and add additional parameters to be tuned.

Data-driven algorithms can be broadly classified into two groups: those that undersample the majority class and those that oversample the minority class. There are also algorithms that combine both processes. Both undersampling and oversampling can be randomly achieved or through a more complicated process of searching for least or most useful instances. Previous works have shown that undersampling the majority class usually leads to better results than oversampling the minority class [2] when oversampling is performed using sampling with replacement from the minority class. Furthermore, combining undersampling of the majority class with oversampling of the minority class has not yielded better results than undersampling of the majority class alone [3]. One of the possible sources of the problematic performance of oversampling is the fact that no new information is introduced in the training set as oversampling must rely on adding new copies of minority class instances already in the data set. Sampling has proven a very efficient method of dealing with class-imbalanced data sets [4], [5].

Removing instances only from the majority class, usually referred to as one-sided selection (OSS) [6], has two major problems. First, reduction is limited by the number of instances of the minority class. Second, instances from the minority class are never removed, even when their contribution to the model's performance is harmful.

However, few attempts have been made to cope with class-imbalanced data sets using instance selection algorithms, which can remove instances from both the minority and majority classes. Standard widely used methods can be applied, but they do not achieve good results because their design bias is not focused on these kinds of problems. Evolutionary computation [7] has been used with more success, but scalability is important [8], and those methods cannot be applied to large and very large data sets.

In this paper, we propose a new framework called *oligarchic* instance selection,[1] which is specifically designed for class-imbalanced data sets. The method has two major objectives: 1) improving the performance of previous approaches based of instances selection for class-imbalanced data sets; and 2) developing a method that is able to scale up to very large, and even huge, problems.

The class-imbalanced nature of the problem is dealt with by means of two mechanisms. First, the selection of instances from the majority and minority classes is performed separately. Second, selection is driven by a *fitness* function that takes accuracy in both classes into account. Furthermore, at its inner level, all the selection process is always performed in balanced sets.

Its divide-and-conquer philosophy addresses the problem of scalability without compromising its performance. The method is based on applying an instance selection algorithm to balanced subsets of the whole training set and on combining the results obtained from those subsets by means of a voting scheme. As an additional and very useful feature, the method has linear time complexity and can be easily implemented in a shared or distributed memory parallel machine.

When dealing with class-imbalanced data sets, our main aim is improving accuracy. However, if a method achieves the same accuracy using less instances, that method would be preferable. Moreover, many of the most relevant class-imbalanced problems appear in very large data sets where data reduction is a must. Thus, in the remaining of this paper, we will focus on both accuracy and reduction.

This paper is organized as follows. Section II presents our proposal, Section III describes our experimental setup, Section IV shows the experimental results, and Section V shows the conclusions of this paper.

[1]We refer to our method as instance selection rather than undersampling because, although the method is specifically designed for class-imbalanced data sets, selection of the minority class is also allowed.

II. OLIGOIS

Although random undersampling works in many cases, the random deletion of examples from the majority class may cause the classifier to miss important concepts pertaining to the majority class [1]. The problem with traditional approaches to class-imbalanced data sets is that they have serious scalability problems, particularly in methods based on evolutionary computation that have been shown to achieve the best performance [9]. Furthermore, when careful experimental comparison is made, many sophisticated methods do not achieve significant improvements over undersampling [9].

As we stated earlier, our goal is to obtain a method that is both scalable and able to sample the most relevant instances to deal with class-imbalanced data sets. Scalability will be achieved using a divide-and-conquer approach. The ability to sample instances to deal with class-imbalanced data sets will be achieved by means of the combination of several rounds of instance selection in balanced subsets of the whole data set.

Thus, our methodology is primarily based on the divide-and-conquer approach. Instead of applying the instance selection method to the whole data set, we first perform a random partition of the instances and apply the selection to each one of the subsets obtained. This partition is repeated for several rounds, and the results are combined through a voting process. To account for the class-imbalanced nature of the data sets, the subsets used always contain the same number of instances from both classes. Any instance selection method can be used in the subset in the same way that any classifier can be used in an ensemble. Because our method treats majority class instances unfairly, favoring minority class instances, we refer to it as *oligarchic* instance selection (OLIGOIS). On its own, each round would not be able to achieve good performance. However, the combination of several rounds using a voting scheme is able to improve the performance of an instance selection algorithm applied to the whole data set with a large reduction in the execution time of the algorithm.

Assume that we have a training set T, with n instances, n^+ from the minority or positive class, and n^- from the majority or negative class. First, the training data T is divided into t disjoint subsets D_j of approximately equal size s as follows:

$$T = \bigcup_{j=1}^{t} D_j. \tag{1}$$

The partition is carried out using a random algorithm. The result of this first step is t subsets where the distribution of classes is roughly as imbalanced as the distribution of the whole data set due to the use of a random algorithm. To avoid effects derived from these uneven distributions, we balance all subsets by adding randomly selected instances of the minority class. These instances are randomly sampled without replacement to avoid repeated instances in any subset. It may happen in heavily imbalanced data sets that there are not enough instances from the minority class to construct balanced data sets. To avoid this from happening, the subset size must fulfill $s \leq 2n^+$.

The instance selection algorithm or a method of our choice is then applied to each subset separately,[2] and the results of all of the subsets are recorded. After applying the instance selection algorithm, we record the number of times that each instance has been selected to be kept. We call this record the number of votes due to its similarity with the combination of classifiers in an ensemble by voting [10]. This process is repeated for r rounds with different random partitions of the data set.

The final step is the combination of the different rounds. To obtain a meaningful combination, we use the philosophy of the ensembles of classifiers. In an ensemble, several *weak* learners are combined to form a strong classifier; in our method, several *weak* (in the sense that they are applied to subsets of the data) instance selection algorithms are combined to produce a strong and fast instance selection method. Each round can be considered similar to a classifier in an ensemble, and the combination process, by voting, is similar to the combination of base learners in bagging or boosting [11].

Once we have performed these r rounds, we have recorded the number of votes received by each instance. The number of votes received by an instance of the majority class is in the interval $[0, r]$, as each majority class instance is only in one subset in every round. Minority class instances may be in more than one subset due to the balance process. Thus, the number of votes is in the interval $[0, t \cdot r]$. The first decision is whether to perform the selection on both classes or only on the majority class. Performing selection only on the majority class constrains the maximum reduction that can be achieved. Furthermore, previous results [9] showed better performance when both classes were subject to selection. Thus, we opted to allow instances to be removed from both classes.

The final combination of votes must set a threshold to decide whether an instance must be selected as the final output of the process. A first natural choice would be majority voting: Instances are kept if they receive at least half of the possible votes. However, the performance of this fixed threshold depends heavily on the problem. Therefore, we developed a method of automatically selecting an optimum vote threshold.

To that end, we define function f that evaluates the goodness of a certain subset of instances S. Using the threshold t, we obtain the set of selected instances $S(t)$, and then $f(S(t))$ is calculated as follows:

$$f\left(S(t)\right) = \alpha r\left(S(t)\right) + (1-\alpha) a\left(S(t)\right) \quad (2)$$

where $r(S(t))$ is the reduction achieved using threshold t to select $S(t)$, and $a(S(t))$ is the accuracy achieved with the instances in $S(t)$ using a 1-nearest neighbor (1-NN) classifier.[3] The accuracy may be calculated using any of the class-imbalanced measures defined in the following. To obtain the best threshold, all of the possible values are evaluated, and the optimum is chosen.

However, this approach does not pay attention to the class-imbalanced nature of the data sets. To account for that, a further improvement is needed. This improvement is achieved using separate thresholds for the minority and majority classes. Two thresholds are then used: t^+ is used for selecting minority class instances and t^- for majority class instances. The evaluation of a pair of thresholds, i.e., t^+ and t^-, is made using the same equation (2), using the subset $S(t^+, t^-)$ selected with these two thresholds. The key difference is that we must evaluate a larger set of values: $[0, r] \times [0, t \cdot r]$. Thus, for each pair of thresholds, we evaluate the following:

$$f\big(S(t^+ \cdot t^-)\big) = \alpha r\left(S(t^+ \cdot t^-)\right) + (1-\alpha) a\left(S(t^+ \cdot t^-)\right). \quad (3)$$

and select the best pair of thresholds. The evaluation of this number of thresholds might exclude the scalability achieved by the divide-and-conquer approach. To avoid this negative effect, the evaluation of a pair of thresholds is also approached using a divide-and-conquer method. Instead of evaluating the accuracy of $S(t^+, t^-)$ with the whole data set, which is of complexity $O(n^2)$, we apply the same partition philosophy used in the previous step. The training set is divided into random disjoint subsets, and accuracy is estimated separately in each subset using the average evaluation of all the subsets for the fitness of each pair of thresholds.

This procedure obtains a selected set of instances that may be imbalanced. To obtain a balanced data set, we perform a last step. The class with more selected instances is undersampled, removing first the instances with fewer votes. This balanced data set is evaluated using (3). If it achieves a better evaluation, the balanced selected data set is used as the final result of the algorithm; otherwise, the selection obtained using the best thresholds is kept. The complete method is shown in Algorithm 1. As OLIGOIS uses a base instance selection to be applied to each subset (see line 2 in Algorithm 1), we will use the term OLIGOIS.X when OLIGOIS is applied using X as instance selection algorithm.

Algorithm 1: OLIGOIS algorithm.

```
   Data: A training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, subset
         size $s$, and number of rounds $r$.
   Result: The set of selected instances $S \subset T$.
   for $i = 1$ to $r$ do
1      Divide instances into $n_s$ disjoint subsets
       $D_i : \bigcup_i D_i = T$ of size $s$
       for $j = 1$ to $n_s$ do
2          Apply instance selection algorithm to $D_j$
3          Store votes of selected instances from $D_j$
       end
   end
4  Obtain thresholds of votes to keep an instance from the
    minority, $t^+$, and the majority, $t^-$, classes
5  $S = \{x_i \in T | (\text{votes}(x_i) \geq t^+ \text{ and } x_i \in C^+) \text{ or } (\text{votes}(x_i) \geq$
    $t^- \text{ and } x_i \in C^-)\}$
6  Undersample the class with more instances in $S$ to obtain
    $S^{\text{balanced}}$ removing instances with fewer votes
    if $f(S^{balanced}) \geq f(S)$ then
        $S = S^{\text{balanced}}$
    end
7   return $S$
```

[2]We can use instance selection methods that are specifically designed for class-imbalanced data sets, or as the subsets are balanced, we can use any standard instance selection method.

[3]Any other classifier can be used in this function to obtain the accuracy of the subset selected.

TABLE I
SUMMARY OF DATASETS. THE ATTRIBUTES OF EACH DATASET CAN BE CONTINUOUS (C), BINARY (B) OR NOMINAL (N). IR IS ALSO SHOWN

	Dataset	Cases	Attributes			Inputs	IR
			C	B	N		
1	abalone19	4177	7	-	1	10	1:130
2	abalone9-18	731	7	-	1	10	1:17
3	ads	3279	-	-	-	1558	1:7
4	adult	48842	6	1	7	105	1:4
5	arabidopsis	33971	-	-	1004	4016	1:124
6	breast-cancer	286	-	3	6	15	1:3
7	cancer	699	9	-	-	9	1:2
8	carG	1728	-	-	6	16	1:25
9	ccds	36,497	-	-	1004	4012	1:26
10	census	29,926	7	-	30	409	1:15
11	dna	50000	-	-	200	800	1:359
12	ecoliM	336	7	-	-	7	1:4
13	ecoliMU	336	7	-	-	7	1:9
14	ecoliOM	336	7	-	-	7	1:16
15	euthyroid	3163	7	18	-	44	1:10
16	german	1000	6	3	11	61	1:3
17	glassBWFP	214	9	-	-	9	1:3
18	glassC	214	9	-	-	9	1:16
19	glassNW	214	9	-	-	9	1:4
20	glassT	214	9	-	-	9	1:23
21	haberman	306	3	-	-	3	1:3
22	hepatitis	155	6	13	-	19	1:4
23	magic04	19,020	10	-	-	10	1:2
24	new-thyroidT	215	5	-	-	5	1:6
25	optdigitsZ	5620	64	-	-	64	1:10
26	ozone1hr	2536	72	-	-	72	1:34
27	ozone8hr	2534	72	-	-	72	1:15
28	pima	768	8	-	-	8	1:2
29	satimageT	6435	36	-	-	36	1:10
30	segmentO	2310	19	-	-	19	1:7
31	sick	3772	7	20	2	33	1:16
32	splice-EI	3175	-	-	60	120	1:4
33	splice-IE	3175	-	-	60	120	1:4
34	titanic	2201	-	-	3	8	1:3
35	ustilago	6141	-	-	1004	4016	1:93
36	vowelZ	990	10	-	-	10	1:11
37	yeastCP	483	8	-	-	8	1:24
38	yeastEXC	1484	8	-	-	8	1:42
39	yeastME1	1484	8	-	-	8	1:33
40	yeastME2	1484	8	-	-	8	1:29

The scalability of the method is assured by the three following features.

1) *Application of the method to small data sets*. Due to the small size of the subsets in which the selection process is applied, the selection process will always be fast, regardless of the complexity of the instance selection method used.
2) *Only small data sets must be kept in memory*. This allows the application of instance selection when data sets do not fit in memory.
3) *Easy parallelization*.

III. EXPERIMENTAL SETUP

We used a set of 40 problems to test the performance of the proposed method, which is shown in Table I. The data sets `ads`, `adult`, `breast − cancer`, `cancer`, `census`, `euthyroid`, `german`, `haberman`, `hepatitis`, `magic04`, `ozone1hr`, `ozone8hr`, `pima`, `sick`, and `titanic` are from the UCI Machine Learning Repository [14]. `arabidopsis` and `ccds` are data sets addressing the problem of recognition of translation initiation sites in gene sequences. `dna` is from the PASCAL challenge.[4] The remaining data sets were created following García *et al.* [15] from UCI data sets. To estimate the reduction and accuracy, we used a tenfold cross-validation method and a 1-NN classifier.

A. *Algorithms for the Comparison*

To allow a fair assessment of the validity of our proposal, we must compare OLIGOIS with state-of-the-art methods for instance selection in class-imbalanced problems. We have chosen the following methods.

1) *Random undersampling*. This method is included as a baseline measure. We randomly undersample the majority class until both classes are balanced. Although more sophisticated methods have been proposed, most experimental results show that this simple method is still very competitive.
2) *Decremental Reduction Optimization Procedure 3 (DROP 3)* [16]. This algorithm is designed to be insensitive to the order of presentation of the instances. It includes a noise filtering step, using a method similar to Wilson's *edited nearest neighbor* rule [17]. Then, the instances are ordered according to distance from their nearest neighbors. Instances are removed beginning with those furthest from their nearest neighbors. This tends to remove the instances furthest from the boundaries first. DROP 3 iteratively removes an instance x if at least as many of its associates in T would be classified correctly without x. The associates of x are the instances that have x as one of their nearest neighbors.
3) *Iterative Case Filtering* (ICF) [18]. For the ICF algorithm, $\text{Coverage}(c) = \{c \in T : c \in \text{LocalSet}(c')\}$ and $\text{Reachable}(c) = \{c' \in T : c' \in \text{LocalSet}(c)\}$ sets are defined. The local set of a case c is defined as "the set of cases contained in the largest hypersphere centered on c such that only cases in the same class as c are contained in the hypersphere" [18]; in other words, the hypersphere is bounded by the first instance of different class. The coverage set of an instance includes the instances that have that instance as one of their neighbors, and the reachable set is formed by the instances that are neighbors to that instance. The algorithm is based on repeatedly applying a deleting rule to the set of retained instances until no more instances fulfill the deleting rule. The deleting rule consists of removing all instances x for which $|\text{Reachable}(x)| > |\text{Coverage}(x)|$.
4) Genetic algorithms have been applied to instance selection. The application is easy and straightforward. Each individual is a binary vector that codes a certain sample of the training set. The evaluation is usually made considering both data reduction and classification accuracy. Examples of applications of genetic algorithms to instance selection can be found in [19]–[21]. Cano *et al.* [22] performed a comprehensive comparison of the performances of different evolutionary algorithms for instance selection. They compared a generational genetic

[4]http://pascallin2.ecs.soton.ac.uk/Challenges/.

algorithm, a steady-state genetic algorithm, a cross generational elitist selection, heterogeneous recombination and cataclysmic mutation (CHC) genetic algorithm, and a population-based incremental learning algorithm. They found that evolutionary-based methods were able to outperform classical algorithms in both classification accuracy and data reduction. Among the evolutionary algorithms, CHC was able to achieve the best overall performance. Therefore, we have included the CHC algorithm in our comparison. To account for the class-imbalanced nature of our data sets, we used G-mean as the accuracy measure instead of a testing error (see Section III-C).

5) García and Herrera [23] performed a comparison of different methods for sampling with class-imbalanced data sets and found evolutionary algorithms to perform earlier nonevolutionary methods, although the computational cost is clearly higher. They proposed a family of different evolutionary methods based on the CHC algorithm in which two different accuracy measures can be used, i.e., G-mean and area under the curve (AUC), and the selection process can be applied to both the minority and majority classes or only to the majority class. The four different combinations were tested with similar results. To allow a fair comparison with the remaining algorithms, we have chosen the G-mean as the accuracy measure and selection of both classes. The authors referred to this method as evolutionary balancing undersampling (EBUS). It uses the following fitness function for an individual S:

$$\text{Fitness}_{\text{Bal}}(S) = \begin{cases} G - |1 - n^+/n^-| \cdot P & \text{if } n^- > 0 \\ G - P & \text{If } n^- = 0 \end{cases} \quad (4)$$

where G is the G-mean, n^+ is the number of instances selected from the minority class, and n^- is the number of instances selected from the majority class. This model aims to remove instances of both classes, identifying minority class examples that have a negative influence over the classification task and achieving maximal reduction in positive instances. A penalty factor P, which is used for preserving the same number of instances belonging to each class, helps to maintain a generalization capability in the reduction task. The authors suggested an empirically obtained value of $P = 0.2$.

6) *Condensed Nearest Neighbor (CNN) Rule* [24]. The standard CNN rule is adapted to class-imbalanced problems. First, we randomly drew one majority class example and all examples from the minority class and put these examples in S. We then tested all of the instances in T not included in S using a 1-NN rule. All misclassified instances are added to S.

7) *OSS* [6]. OSS is an undersampling method resulting from the application of Tomek links (TLs) followed by the application of CNN. TLs [25] are defined as follows: Given a pair of instances x_i and x_j, belonging to classes y_i and y_j, respectively, where $y_i \neq y_j$ and $d(x_i, x_j)$ is the distance between them, a pair (x_i, x_j) is called a TL if there is not an example x_l such that $d(x_i, x_l) < d(x_i, x_j)$ or $d(x_j, x_l) < d(x_i, x_j)$. If two instances form a TL, either one of them is noise or both are borderline samples. TLs can be used as an undersampling method, removing for every TL found the instance belonging to the majority class.

8) $CNN + TL$ [26]. This method is similar to OSS, but the CNN rule is applied before the TLs. The method is faster than OSS because TLs are obtained for a smaller data set.

9) *Modified Selective Subset* (MSS) [27] *method*. This method is a modification of the algorithm of Ritter *et al.* [28] for finding a selective subset. A subset is selective if it is consistent, and all prototypes in the original training set are nearer to a selective neighbor of the same class than to any member of the training set from a different class. The MSS algorithm is aimed at obtaining a minimal consistent subset using the selective property. In this way, the authors define the MSS as that subset of the training set that contains, for every instance x in the training set, that element of its neighborhood that is the nearest to a class other than that of x. The authors propose an iterative procedure to find this modified selective subset. Each instance is ordered regarding the distance to its nearest enemy.

10) Synthetic minority over-sampling technique (SMOTE) method [2]. One of the problems with oversampling is that merely making copies of the minority class samples does not add new information to the data set. To overcome this problem, Chawla *et al.* [2] proposed a method called SMOTE, which combines undersampling of the majority class with oversampling of the minority class. However, instead of oversampling the minority class by just making copies of the minority class samples, SMOTE generates synthetic instances from actual instances of the minority class. Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration.

The source code used for all methods is in C and is licensed under the GNU General Public License. The code, the partitions of the data sets, and the detailed numerical results of all the experiments are available from http://cib.uco.es/index.php/supplementary-material-for-oligois.

All of the experiments were carried out in a cluster of 32 blades. Each blade is a biprocessor DELL Power Edge M600 with four cores per processor. Thus, we count 256 cores. The blades are interconnected with a master node and among them with a 1-Gb network. In the parallel implementation, we use a master/slave model in which all information processed by the slaves is sent by the master. The processors run at 2.5 GHz, and each blade has 16 Gb of memory.

B. Statistical Tests

We used the Wilcoxon test as the main statistical test for comparing pairs of algorithms. This test was chosen because it assumes limited commensurability and is safer than parametric tests because it does not assume normal distributions or homogeneity of variance. Therefore, it can be applied to accuracy and reduction. Furthermore, empirical results [29] show that it is also stronger than other tests.

For groups of methods, we first carry out an Iman–Davenport test to ascertain whether there are significant differences among methods. The Iman–Davenport test is based on the χ_F^2 Friedman test, which compares the average ranks of k algorithms, but is more powerful than the Friedman test. Let r_i^j be the rank of the jth algorithm on the ith data set, where in case of ties, average ranks are assigned, and let $R_j = 1/N \sum_i r_i^j$ be the average rank for N data sets. Under the null hypothesis, all algorithms are equivalent; the statistic

$$X_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{5}$$

is distributed following χ_F^2 with $k-1$ degrees of freedom for k and N sufficiently large. In general, $N > 10$ and $k > 5$ is enough. Iman and Davenport found this statistic to be too conservative and developed a better one as follows:

$$F_F = \frac{(N-1)X_F^2}{N(k-1) - X_F^2} \tag{6}$$

which is distributed following a F distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. After carrying out the Iman–Davenport test, we can perform pairwise comparisons with the Wilcoxon test. However, when all algorithms are compared with a control method, it is not advisable to perform many Wilcoxon tests against the control method. We can instead use one of the general procedures for controlling the familywise error in multiple hypothesis testing. The test statistic for comparing the ith and jth classifiers using these methods is as follows:

$$z = \frac{(R_i - R_j)}{\sqrt{k(k+1)/6N}}. \tag{7}$$

The z-value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate α. Step-up and step-down procedures sequentially test the hypotheses ordered by their significance. We will denote the ordered p-values by $p_1, p_2, \ldots$ so that $p_1 \leq p_2 \leq \ldots \leq p_{k-1}$. One of the simplest such methods was developed by Holm. It compares each p_i with $\alpha/(k-i)$. Holm's step-down procedure starts with the most significant p value. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected, and we are allowed to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all remaining hypotheses are retained as well. We will use for all statistical tests a significance level of 0.05.

C. Evaluation Measures

Accuracy is not a useful measure for imbalanced data, particularly when the number of instances of the minority class is very small compared with the majority class. If we have a ratio of 1 : 100, a classifier that assigns all instances to the majority class will have 99% accuracy. Several measures have been developed to take the imbalanced nature of the problems into account. Given the number of true positives (TPs), false positives (FPs), true negatives (TNs), and false negatives (FNs), we can define several measures. Perhaps, the most common measures are the TP rate TP_{rate}, recall R, or sensitivity Sn, i.e.,

$$TP_{\text{rate}} = R = Sn = \frac{TP}{TP + FN} \tag{8}$$

which is relevant if we are only interested in the performance on the positive class and the TN rate TN_{rate} or specificity Sp, as follows:

$$TN_{\text{rate}} = Sp = \frac{TN}{TN + FP}. \tag{9}$$

From these basic measures, others have been proposed, such as the F-measure or, if we are concerned about the performance of both negative and positive classes, the G-mean measure: $G-\text{mean} = \sqrt{Sp \cdot Sn}$. This last measure is used as the accuracy measure in all of our experiments. Thus, in the tables reporting our results, we will show the G-mean measure and the reduction after applying the instance selection algorithm. Reduction is measured as the percentage of instances removed after applying the selection process.

Many classifiers are subject to some kind of threshold that can be varied to achieve different values of the earlier measures. For that kind of classifiers, receiver operating characteristic (ROC) curves can be constructed. An ROC curve is a graphical plot of the TP_{rate} (sensitivity) against the FP_{rate} $(1-$ specificity or $FP_{\text{rate}} = FP/TN + FP)$ for a binary classifier system as its discrimination threshold is varied. The perfect model would achieve a TP rate of 1 and an FP rate of 0. A random guess will be represented by a line connecting the points (0, 0) and (1, 1). ROC curves are a good measure of the performance of the classifiers. Furthermore, from this curve, a new measure, i.e., AUC, can be obtained, which is a very good overall measure for comparing algorithms. AUC is a useful metric for classifier performance as it is independent of the decision criterion selected and prior probabilities.

However, for the nearest neighbor rule, AUC is less commonly used due to the difficulty of obtaining the threshold needed to construct the ROC curve. We will present AUC results for the nearest neighbor using as threshold the ratio between the distances to the nearest neighbor for each one of the two classes.

IV. Experimental Results

As stated, OLIGOIS is a method that can use any of the many available base instance selection algorithms, in the same way that boosting can use the classifier of choice. Thus, the first set of experiments focused on determining which instance selection algorithm was most appropriate for OLIGOIS. We performed experiments using OLIGOIS with eight of the nine methods described earlier. Random undersampling was not used because the subsets used in our method are always balanced.

Tables II and III show a pairwise comparison of all the methods for accuracy and reduction, respectively. Fig. 1 shows

TABLE II
PAIRWISE COMPARISON OF OLIGOIS ACCURACY MEASURED USING G-MEAN USING THE EIGHT METHODS. WIN/LOSS RECORD AND THE p-VALUE OF THE WILCOXON TEST ARE SHOWN

	OLIGOIS +							
	CHC	CNN	CNN+TL	DROP3	EBUS	ICF	MSS	OSS
Mean	0.8221	0.6235	0.7326	0.6609	0.7914	0.6361	0.6620	0.7271
CHC		3/37	5/35	3/37	9/31	0/40	0/40	7/33
		0.0000	0.0000	0.0000	0.0002	0.0000	0.0000	0.0000
CNN			31/9	21/19	36/4	20/20	22/18	31/9
			0.0002	0.1393	0.0000	0.6380	0.1393	0.0002
CNN+TL				9/31	30/10	8/32	10/30	24/16
				0.0028	0.0013	0.0001	0.0035	0.9357
DROP3					33/7	19/21	22/18	30/10
					0.0000	0.5101	0.3897	0.0026
EBUS						5/35	6/34	13/27
						0.0000	0.0000	0.0042
ICF							22/18	30/10
							0.0983	0.0003
MSS								28/12
								0.0039
Iman-Davenport p -value: 0.0000								

TABLE III
PAIRWISE COMPARISON OF OLIGOIS FOR REDUCTION MEASURED AS THE PERCENTAGE OF INSTANCES REMOVED USING THE EIGHT METHODS. WIN/LOSS RECORD AND THE p-VALUE OF THE WILCOXON TEST ARE SHOWN

	OLIGOIS +							
	CHC	CNN	CNN+TL	DROP3	EBUS	ICF	MSS	OSS
Mean	0.9658	0.9328	0.9408	0.9189	0.9563	0.9135	0.9112	0.9022
CHC		9/31	8/32	4/35	10/30	2/38	4/36	1/39
		0.0001	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000
CNN			20/20	18/22	27/13	16/24	19/21	13/27
			0.4278	0.1789	0.0015	0.0385	0.1878	0.0111
CNN+TL				15/25	27/13	12/28	14/26	10/30
				0.0069	0.0161	0.0023	0.0056	0.0015
DROP3					33/7	15/25	22/18	14/26
					0.0000	0.1923	0.7470	0.1039
EBUS						4/36	6/34	4/36
						0.0000	0.0000	0.0000
ICF							23/17	20/19
							0.4846	0.5678
MSS								15/25
								0.1878
Iman-Davenport p -value: 0.0000								

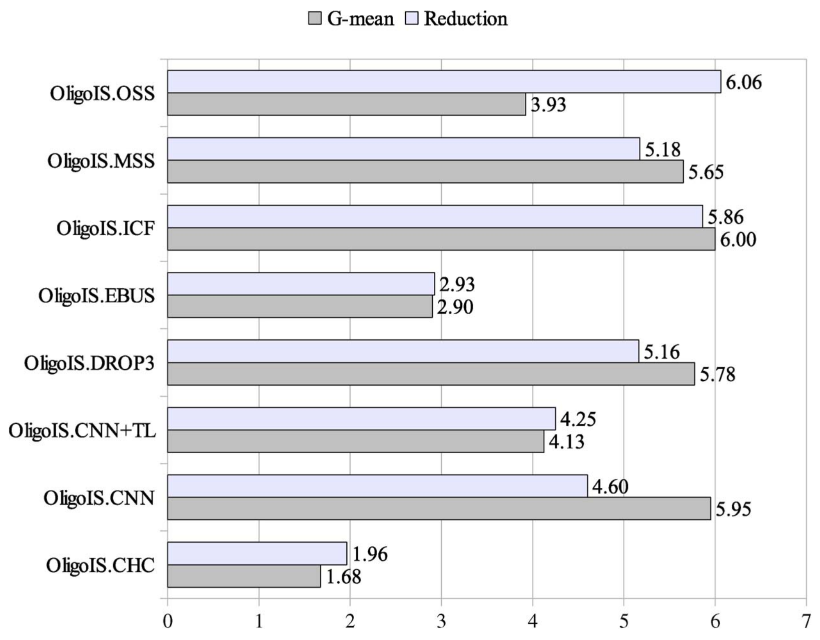

Fig. 1. Average Friedman's ranks for OLIGOIS using the eight instance selection algorithms.

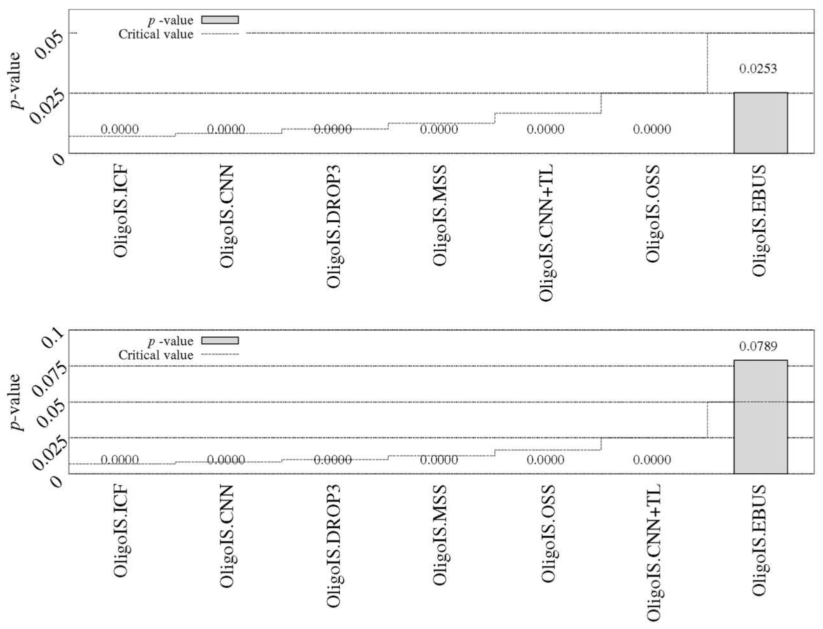

Fig. 2. Results of the Holm test for OLIGOIS with seven different methods and OLIGOIS.CHC as the control method for (top) accuracy and (bottom) reduction. Numerical p-values are shown.

average Friedman rankings for the same experiments. These ranks are, by themselves, a good measure of the relative performance of a group of methods. The Iman–Davenport test with a p-value of 0 for both accuracy and reduction showed significant differences between the methods. The rankings showed that the best performing method was OLIGOIS.CHC. The pairwise comparisons also showed that this method was significantly better than all the others, with a p-value below 0.05 in both accuracy and reduction. Results for OLIGOIS.EBUS were also good but inferior to OLIGOIS.CHC.

However, as stated in Section III-B, using many pairwise comparisons is not the most effective way of comparing a number of algorithms. To further assure the advantage of OLIGOIS.CHC, we used Holm's procedure. OLIGOIS.CHC was used as the control method. Fig. 2 shows the results of Holm's test for accuracy and reduction. The test showed that OLIGOIS.CHC was better than all other methods in terms of accuracy and better than all the methods except OLIGOIS.EBUS in terms of reduction. Thus, we chose OLIGOIS.CHC as representative of OLIGOIS to be used in the remaining experiments.

In the previous experiments, we compared OLIGOIS using eight different instance selection methods. However, we have not compared the performance of OLIGOIS against its base method. Therefore, we compared each application of OLIGOIS for each instance selection algorithm with the application of the same instance selection algorithm to the whole data set. We used the Wilcoxon test to compare pairs of algorithms.

Table IV shows the results of the Wilcoxon test. The table shows that OLIGOIS improved all of the methods in terms of reduction and some of them in terms of both reduction and accuracy. It is particularly important to note that OLIGOIS was better than the standard forms of the two best performing methods CHC and EBUS. For EBUS, it improved reduction while preserving accuracy, and for CHC, it improved reduction and accuracy. Although for some methods, accuracy was equal to or worse than the standard method, accuracy is exchanged for a significant improvement in reduction. Furthermore, performance using those methods is less relevant because the representative of our methodology is OLIGOIS.CHC, which was shown to be an improvement on all of the methods in the next experiment performed.

TABLE IV
p-VALUES OF THE PAIRWISE COMPARISON USING A WILCOXON TEST OF OLIGOIS USING THE EIGHT METHODS APPLIED IN THEIR STANDARD WAYS. A ● MEANS THAT OUR METHOD IS SIGNIFICANTLY BETTER, A ○ THAT IT IS WORSE, AND A — THAT THERE ARE NO DIFFERENCES

Method	OLIGOIS		
	G-mean	AUC	Reduction
CHC	0.0000●	0.0000●	0.0000●
CNN	0.0008○	0.0252○	0.0000●
CNN+TL	0.0360●	0.9732−	0.0000●
DROP3	0.3265−	0.0016●	0.0144●
EBUS	0.2113−	0.1562−	0.0000●
ICF	0.0231○	0.0007●	0.0000●
MSS	0.6190−	0.0000○	0.0000●
OSS	0.6190−	0.7726−	0.0000●

Once we had established the utility of OLIGOIS and chosen OLIGOIS.CHC as its representative, we compared OLIGOIS.CHC with the ten state-of-the-art methods described in Section III-A. All of the algorithms were applied to the whole data set with the same parameters used for OLIGOIS. Table V shows the results of OLIGOIS.CHC and the ten standard methods for AUC and reduction.

We performed a Holm test using our proposed method as the control algorithm. Before applying the Holm test, we ascertained global differences with an Iman–Davenport test that obtained a p-value of 0 for both accuracy and reduction.

The results of the Holm test are shown in Fig. 3. In terms of both accuracy and reduction, OLIGOIS.CHC showed a significantly better performance than all of the other standard

TABLE V
ACCURACY AND REDUCTION OF THE TEN STANDARD METHODS AND OLIGOIS.CHC

Dataset	OLIGOIS.CHC		CHC		CNN		CNN+TL		DROP3		EBUS		ICF		MSS		OSS		SMOTE		Undersampling	
	Reduction	AUC	Reduction	AUC	Reduction	AUC	Reduction	AUC	Reduction	AUC	Reduction	AUC	Reduction	AUC	Reduction	AUC	Reduction	AUC	Reduction	AUC	Reduction	AUC
abalone19	0.9911	0.9012	0.8268	0.7910	0.9409	0.7034	0.9668	0.6934	0.9623	0.5277	0.8858	0.9847	0.9605	0.5395	0.9477	0.5788	0.9453	0.6736	0.9540	0.8027	0.9847	0.8055
abalone9-18	0.9726	0.8671	0.9042	0.8437	0.8091	0.7946	0.8947	0.8232	0.8681	0.6881	0.8711	0.9965	0.8401	0.7136	0.8386	0.7819	0.8343	0.8505	0.6553	0.7787	0.8851	0.7814
ads	0.9480	0.9175	0.8397	0.9167	0.7893	0.7159	0.8308	0.9074	0.9255	0.8809	0.8699	0.9228	0.8406	0.9018	0.8906	0.9066	0.7982	0.7609	0.1601	0.5002	0.7200	0.7781
adult	0.9770	0.8232	0.7744	0.8089	0.5565	0.7309	0.7139	0.8106	0.7111	0.7425	0.9004	0.8099	0.6705	0.7297	0.6676	0.7459	0.6450	0.8046	-0.4357	0.7543	0.5214	0.8122
arabidopsis	0.9899	0.9385	0.7646	0.6816	0.3289	0.5792	0.7815	0.6609	0.6302	0.5554	0.8985	0.6367	0.6430	0.5550	0.5976	0.5876	0.4051	0.6742	0.9518	0.6805	0.9839	0.6396
breast-cancer	0.9666	0.7152	0.9066	0.6667	0.3624	0.5786	0.6473	0.5758	0.5310	0.5631	0.7605	0.6143	0.3675	0.5579	0.4930	0.6401	0.5081	0.6112	-0.5953	0.5294	0.4046	0.5866
cancer	0.9803	0.9422	0.9859	0.9506	0.6149	0.9408	0.6479	0.9568	0.9095	0.9541	0.8314	0.9504	0.8905	0.9534	0.8818	0.9480	0.6349	0.9573	-0.6902	0.9534	0.3099	0.9544
carG	0.9612	0.9952	0.8727	0.9886	0.9039	0.9584	0.9428	0.9857	0.9274	0.9658	0.8515	0.9801	0.8594	0.9789	0.8981	0.9321	0.9093	0.9947	0.7604	0.9951	0.9202	0.9863
ccds	0.9810	0.7549	0.7667	0.7105	0.7384	0.6593	0.8728	0.7137	0.8538	0.6735	0.9003	0.6739	0.8965	0.6726	0.8224	0.6813	0.7616	0.7100	0.7710	0.7336	0.9237	0.7165
census	0.9747	0.9335	0.7670	0.7769	0.8047	0.7050	0.8894	0.7855	0.8665	0.6277	0.9002	0.7855	0.8380	0.5939	0.8267	0.7468	0.8304	0.7627	0.6167	0.8178	0.8722	0.8045
dna	0.9940	0.8219	0.7679	0.7071	0.9398	0.7216	0.9742	0.7183	0.9629	0.6263	0.8993	0.7360	0.9706	0.6176	0.9394	0.7426	0.9412	0.7307	0.9833	0.7481	0.9944	0.7147
ecoliM	0.9419	0.9526	0.9584	0.9268	0.6311	0.8753	0.7327	0.9098	0.8017	0.9220	0.7898	0.9302	0.7888	0.8921	0.7525	0.9026	0.6875	0.9312	-0.3743	0.9345	0.5420	0.9262
ecoliMU	0.9657	0.9400	0.9716	0.9367	0.7723	0.8237	0.8637	0.9301	0.8383	0.8887	0.8522	0.8637	0.8235	0.8650	0.8004	0.8646	0.8175	0.9480	0.3743	0.9119	0.7914	0.9428
ecoliOM	0.9597	0.9608	0.9755	0.9688	0.8912	0.9463	0.9206	0.9816	0.9537	0.9516	0.9068	0.9830	0.9448	0.9323	0.9319	0.9526	0.8998	0.9458	0.6429	0.9825	0.8810	0.9937
euthyroid	0.9423	0.9357	0.8282	0.8726	0.7908	0.8613	0.8642	0.8975	0.8752	0.8073	0.8715	0.9089	0.8156	0.8340	0.8245	0.8418	0.8125	0.9007	0.4443	0.9116	0.8147	0.9014
german	0.9757	0.7412	0.8648	0.6981	0.3637	0.6323	0.6295	0.6552	0.5518	0.6681	0.8239	0.6972	0.5553	0.6581	0.4968	0.6761	0.4976	0.6879	-0.6000	0.6966	0.4000	0.6790
glassBWFP	0.9197	0.9326	0.9130	0.8539	0.4311	0.8393	0.6067	0.9097	0.6948	0.8706	0.6855	0.9191	0.6627	0.8592	0.6166	0.8675	0.5161	0.8991	-0.6528	0.8983	0.3472	0.8959
glassC	0.9299	0.9762	0.9621	0.9912	0.8671	0.9700	0.9159	0.9750	0.9232	0.9837	0.8852	0.9860	0.8727	0.9937	0.8956	0.9875	0.8884	0.9925	0.6355	0.9950	0.8785	0.9625
glassNW	0.9378	0.9879	0.9554	0.9846	0.6933	0.9640	0.7441	0.9899	0.8912	0.9765	0.7679	0.9783	0.8741	0.9786	0.8508	0.9895	0.7150	0.9900	-0.4285	0.9626	0.5233	0.9870
glassT	0.9112	0.9880	0.9631	0.9952	0.8883	0.9927	0.9283	0.9833	0.9377	0.9754	0.9169	0.7973	0.8940	0.9682	0.9091	0.9831	0.8982	0.9975	0.7506	0.9928	0.9169	0.9636
haberman	0.9420	0.8468	0.8986	0.6558	0.3931	0.6624	0.6819	0.6698	0.5554	0.6753	0.7424	0.6741	0.4040	0.6908	0.5112	0.6088	0.5757	0.6763	-0.5297	0.6003	0.4703	0.6552
hepatitis	0.9400	0.9988	0.9364	0.8944	0.5507	0.7863	0.7550	0.8737	0.6893	0.8470	0.7728	0.8309	0.6464	0.8250	0.6286	0.8557	0.6350	0.8753	-0.2300	0.8840	0.5900	0.9029
magic04	0.9791	0.8316	0.7967	0.8319	0.4436	0.7957	0.6023	0.8417	0.6957	0.8417	0.9011	0.8278	0.6750	0.8323	0.6376	0.8349	0.5144	0.8565	-0.7033	0.8563	0.2968	0.8623
new-thyroidT	0.9588	0.9763	0.9748	0.9993	0.7892	0.9974	0.8196	0.9991	0.9392	0.9798	0.8206	0.9800	0.9103	0.9853	0.9278	0.9908	0.7985	0.9947	0.0227	1.0000	0.6742	1.0000
optdigitsZ	0.9836	0.9507	0.8529	0.9566	0.8829	0.9566	0.8923	0.9566	0.9894	0.9566	0.8871	0.9567	0.9878	0.9566	0.9585	0.9566	0.8830	0.9566	0.4085	0.9566	0.8029	0.9566
ozone1hr	0.9714	0.9020	0.8241	0.8865	0.8620	0.7780	0.9263	0.8328	0.9074	0.7680	0.8751	0.9013	0.9121	0.7473	0.8612	0.7952	0.8780	0.8443	0.8273	0.8309	0.9425	0.8546
ozone8hr	0.9684	0.9095	0.8312	0.8476	0.7816	0.7828	0.8808	0.8323	0.8530	0.7762	0.8732	0.8403	0.8600	0.7509	0.8025	0.7942	0.8106	0.8495	0.6210	0.8376	0.8737	0.8436
pima	0.9743	0.8172	0.8877	0.7674	0.3623	0.6646	0.5990	0.7348	0.5681	0.7509	0.7935	0.7910	0.5400	0.7421	0.5179	0.6859	0.4916	0.7548	-0.6974	0.7242	0.3026	0.7502
satimageT	0.9659	0.9088	0.8133	0.9124	0.7917	0.9147	0.8680	0.9220	0.8843	0.8872	0.8891	0.9141	0.8910	0.8723	0.8310	0.9200	0.8182	0.9305	0.4164	0.9300	0.8055	0.9218
segmentO	0.9692	0.9483	0.8907	0.9581	0.8228	0.9583	0.8418	0.9591	0.9736	0.9584	0.8695	0.9607	0.9368	0.9583	0.9419	0.9587	0.8249	0.9591	0.1429	0.9591	0.7143	0.9590
sick	0.9778	0.9440	0.8214	0.8927	0.8556	0.8728	0.9074	0.8995	0.9101	0.8163	0.8803	0.9054	0.8796	0.8123	0.8645	0.8872	0.8687	0.9090	0.6326	0.9136	0.8775	0.9099
splice-EI	0.9572	0.9167	0.8361	0.8496	0.4399	0.8584	0.6851	0.8964	0.6905	0.8661	0.8727	0.8651	0.6484	0.8748	0.6152	0.8863	0.5318	0.9060	-0.4400	0.9227	0.5200	0.9024
splice-IE	0.9589	0.8387	0.8277	0.8275	0.4535	0.8421	0.6884	0.8846	0.6786	0.8189	0.8733	0.8423	0.6288	0.8384	0.6033	0.8593	0.5427	0.8892	-0.4456	0.9015	0.5181	0.8918
titanic	0.9828	0.6660	0.8758	0.0021	0.0270	0.0000	0.6769	0.4662	0.9948	0.4263	0.8702	0.0021	0.9990	0.2462	0.9990	0.5897	0.6747	0.4092	-0.6461	0.0000	0.3540	0.0000
ustilago	0.9880	0.9930	0.8338	0.6931	0.0000	0.6881	0.0000	0.6792	0.9588	0.7233	0.8884	0.7777	0.9617	0.7527	0.9584	0.7342	0.0000	0.6336	0.9365	0.7504	0.9789	0.6616
vowelZ	0.9738	0.9959	0.9201	0.9825	0.8836	0.9496	0.8996	0.9817	0.9732	0.9814	0.8659	1.0167	0.8961	0.9818	0.9261	0.9851	0.8836	0.9939	0.4545	0.9956	0.8182	0.9948
yeastCP	0.9618	0.9900	0.9593	0.8846	0.8526	0.8228	0.9162	0.8964	0.9089	0.8607	0.9319	0.9692	0.8919	0.8578	0.8609	0.8671	0.8648	0.8804	0.7516	0.8224	0.9172	0.8997
yeastEXC	0.9870	0.9760	0.9013	0.9480	0.9052	0.8732	0.9449	0.9315	0.9406	0.8446	0.8670	0.9430	0.9424	0.7754	0.9193	0.8578	0.9153	0.9175	0.8585	0.9433	0.9529	0.9477
yeastME1	0.9852	0.9916	0.9046	0.9873	0.9326	0.9778	0.9564	0.9904	0.9594	0.9418	0.8728	0.9908	0.9612	0.8323	0.9404	0.9694	0.9402	0.9808	0.8222	0.9718	0.9407	0.9903
yeastME2	0.9850	0.9244	0.8689	0.9102	0.8680	0.8228	0.9294	0.9128	0.9078	0.7622	0.8555	0.8745	0.9010	0.7276	0.8651	0.8265	0.8841	0.8719	0.7939	0.9096	0.9313	0.8420
Avearage	0.9658	0.9088	0.8756	0.8439	0.6754	0.7999	0.7960	0.8506	0.8398	0.8083	0.8568	0.8504	0.8120	0.7964	0.8013	0.8305	0.7320	0.8478	0.2230	0.8322	0.7224	0.8395

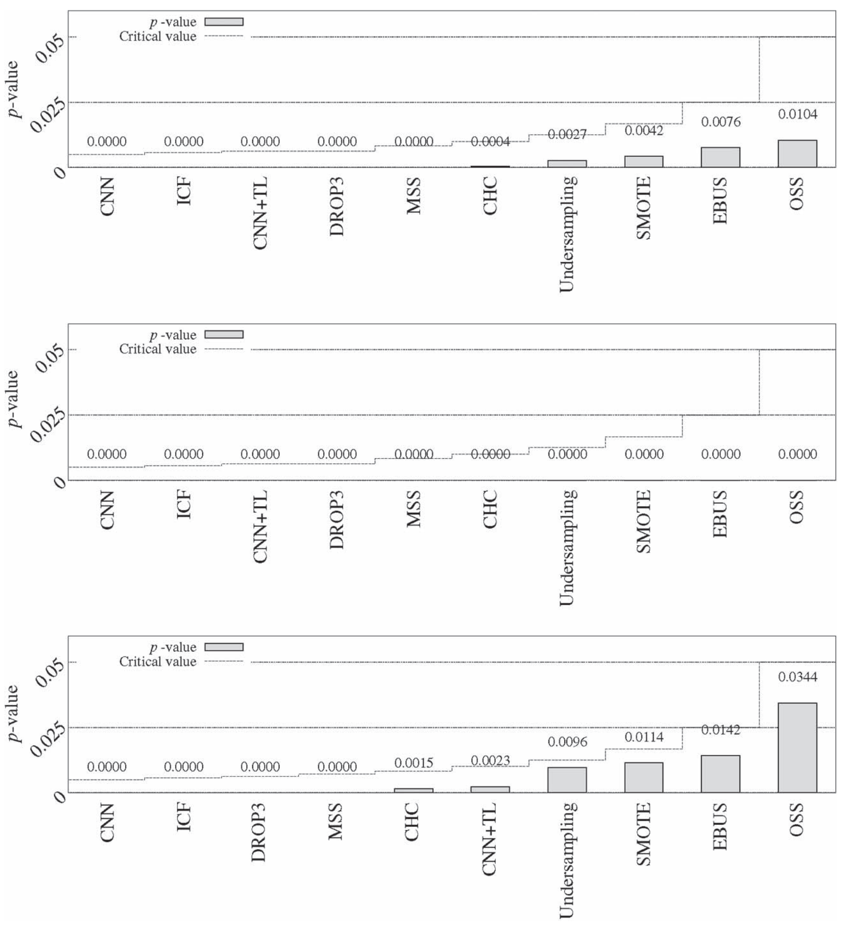

Fig. 3. Results of the Holm test for the standard methods and OLIGOIS.CHC as the control method for (top) G-mean, (middle) reduction, and (bottom) AUC. Numerical p-values are shown.

methods. A plot of average rank values is shown in Fig. 4 together with the average values of AUC, G-mean, and reduction for all methods. This plot shows that average ranks are clearly better for OLIGOIS.CHC. The figure also shows that OLIGOIS.CHC not only achieved superior accuracy to the remaining methods but that reduction was also clearly better than the other nine methods.

The results of the nine methods against OLIGOIS.CHC are illustrated in Fig. 5. The figure shows results for accuracy and reduction. This graphic representation is based on the κ-error relative movement diagrams [30]. However, here, we use the reduction difference instead of the κ difference value. These diagrams use an arrow to represent the results of two methods applied to the same data set. The arrow starts at the

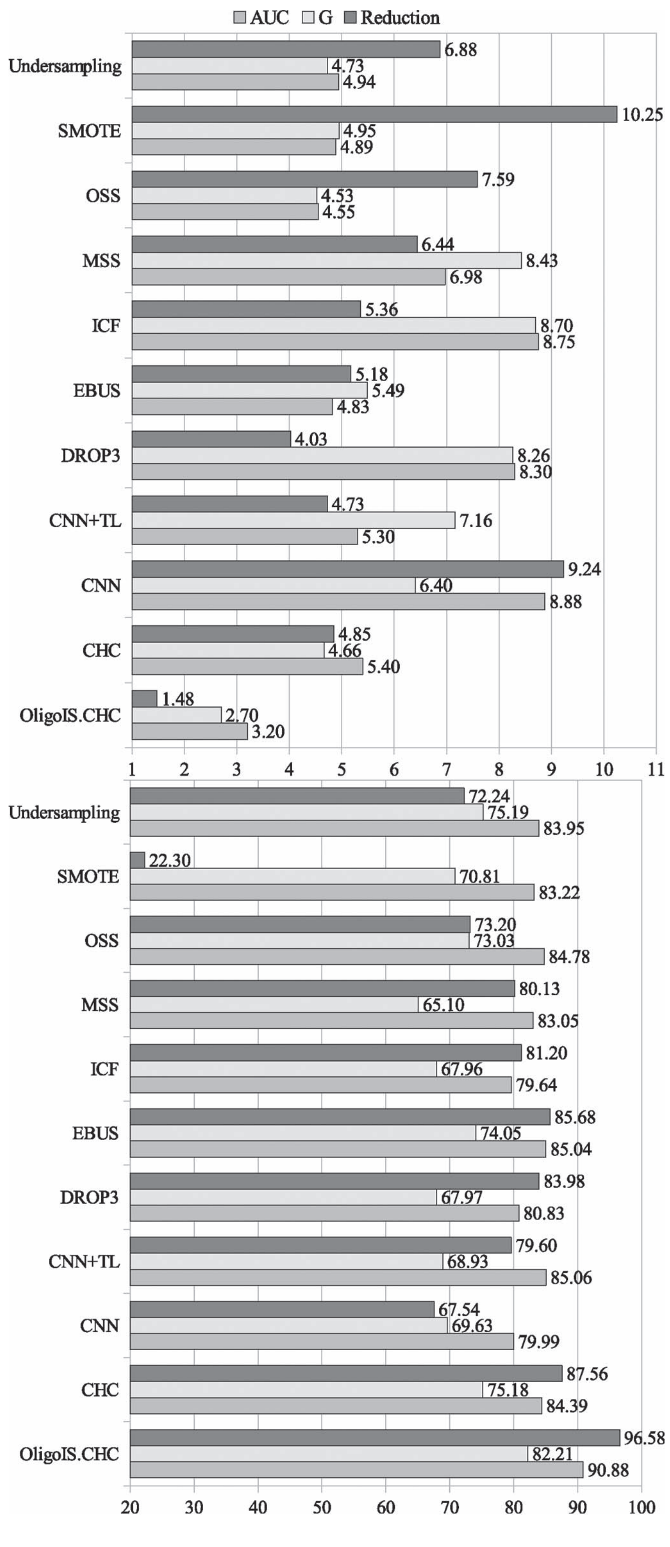

Fig. 4. Average Friedman's ranks and average values for the standard algorithms and OLIGOIS.CHC.

coordinate origin, and the coordinates of the tip of the arrow represent the difference between the accuracy and reduction of our method and those of the standard undersampling algorithm. These graphs are a convenient way to summarize the results. A positive value in either reduction or accuracy means that our method performed better. Thus, arrows pointing up-right represent data sets for which our method outperformed the standard algorithm in both accuracy and reduction. Arrows pointing up-left indicate that our algorithm improved reduction but had worse accuracy, whereas arrows pointing down-right indicate that our algorithm improved accuracy but had a worse reduction. Arrows pointing down-left indicate that our algorithm performed worse in both accuracy and reduction.

If we inspect the plots, we can see that most of the arrows are in the top right part of the coordinates, meaning a general advantage of OLIGOIS.CHC in both accuracy and reduction. We can also see a remarkable improvement in some problems.

One of the most interesting features of any method of undersampling or instance selection for class-imbalanced data sets is its behavior when the imbalance ratio (IR) increases. It is important that a method is able to keep its performance for high IRs as well as for more balanced data sets. To test the performance of OLIGOIS.CHC when the IR increases, we show in Fig. 6 the accuracy and reduction of OLIGOIS.CHC and the second best performing method, CHC, for the ten data sets with the highest IRs.

The figure shows that the behavior of OLIGOIS.CHC for high IRs was even better than in the general case. It clearly achieved a better AUC than CHC and, at the same time, showed a reduction that was, on average, 15% larger than the reduction obtained by CHC. The differences are significant for accuracy and reduction using a Wilcoxon test and a significance level of 0.05.

A. Execution Time

In the description of our method, we claimed that applying the philosophy of divide-and-conquer would produce a method that was more scalable than previous approaches. To verify this claim, Fig. 7 shows the average time of each method together with the time needed for the longest execution. The execution time shown in the tables is the wall-clock time spent by each part of the algorithm. We measure the time elapsed from the beginning of the algorithm until it completes its final output. That means that the time needed to read the data files, perform the partition, send the data to the slaves, receive the results, and obtain the best threshold of votes is included in the reported time. In this way, our proposal establishes no artificial advantage. The figure shows that OLIGOIS.CHC is significantly faster than the other methods. The difference is particularly remarkable for the largest problems and the parallel implementation of our method. Although we are comparing a parallel implementation of OLIGOIS.CHC against the other methods, we do not think that it is an unfair comparison as natural parallelization is an inherent feature of OLIGOIS.

B. Study of the Method

The presented method has three parameters that must be set by the user. Here, we study the behavior of OLIGOIS.CHC as regards these three parameters. For any method, high sensitivity to any of its parameters is undesirable, as this would make difficult the use of the method. The first parameter is the size of the subsets. We chose a size of 100 instances for our previous

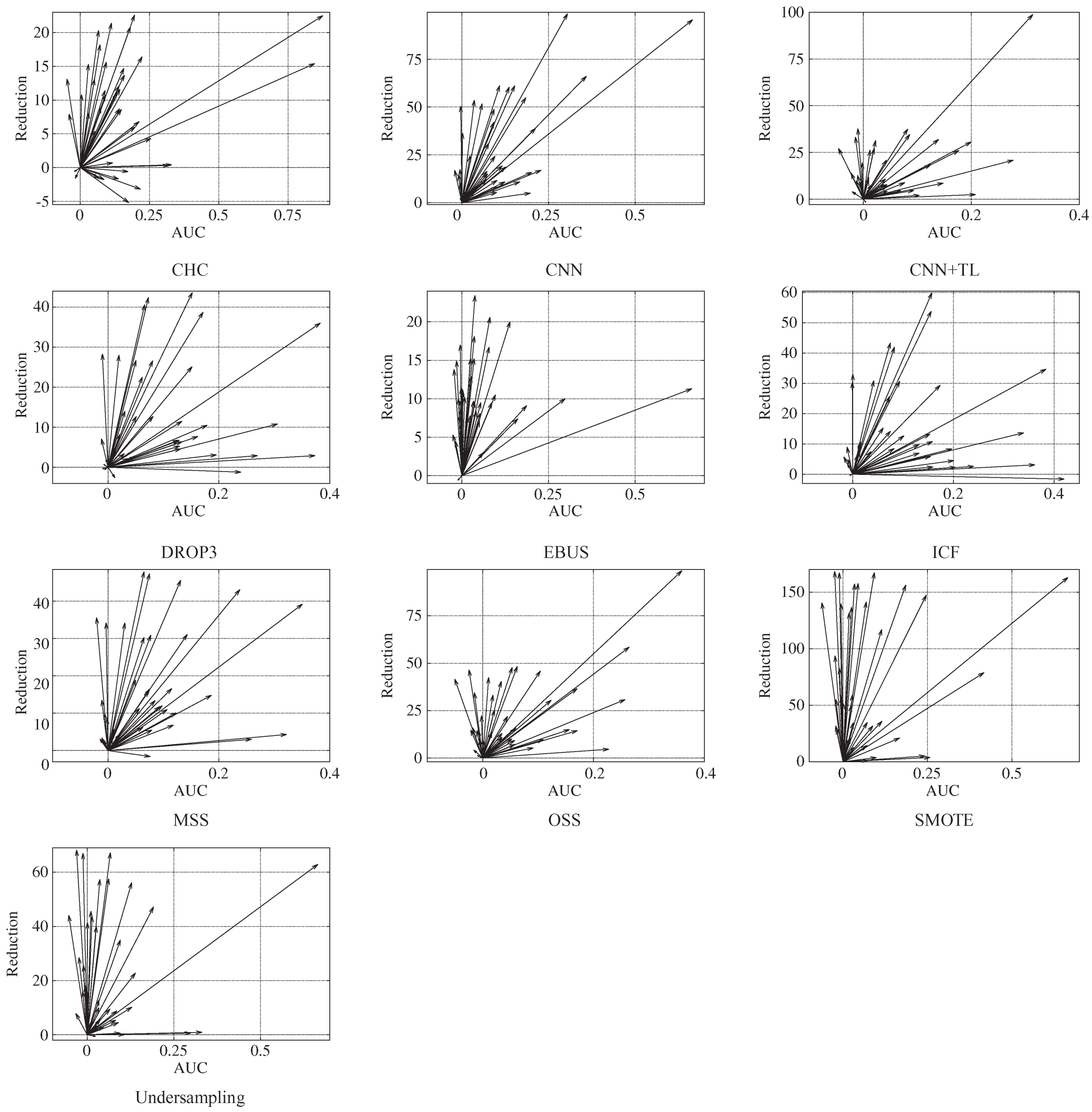

Fig. 5. Reduction/accuracy using relative movement diagrams for OligoIS.CHC and the nine standard methods. Positive values on both axes show better performance by OligoIS.CHC.

experiments. This size achieved good performance and fast execution. In the first experiment, we tested whether a larger size would achieve better results. Fig. 8 shows the average accuracy, reduction, and execution time for subset sizes of 100, 250, 500, 750, and 1000 instances.

The results showed that the method is quite stable with respect to this parameter. An Iman–Davenport test failed to find significant differences among the different values of the parameter in terms of accuracy, with a p-value of 0.4883. For reduction, the Iman–Davenport test obtained a p-value of 0.0015, showing significant differences. We performed a Holm test using the value of 100 instances as a control experiment because this parameter obtained the best average rank. The results of the Holm test are shown in Fig. 9. These results show that the best subset size in terms of reduction is 100 instances.

The second relevant parameter of our method is the number of rounds. We chose ten rounds, using the results of other approaches combining difference methods, such as ensembles of classifiers, which have shown that most of the performance gain is obtained by the first few classifiers added. To test the validity of this value, we performed experiments with 5, 10, 15, 20, and 25 rounds. Fig. 10 shows the average accuracy, reduction, and execution time of the experiments conducted on each number of rounds. As in the previous experiment, we performed an Iman–Davenport test for accuracy and reduction.

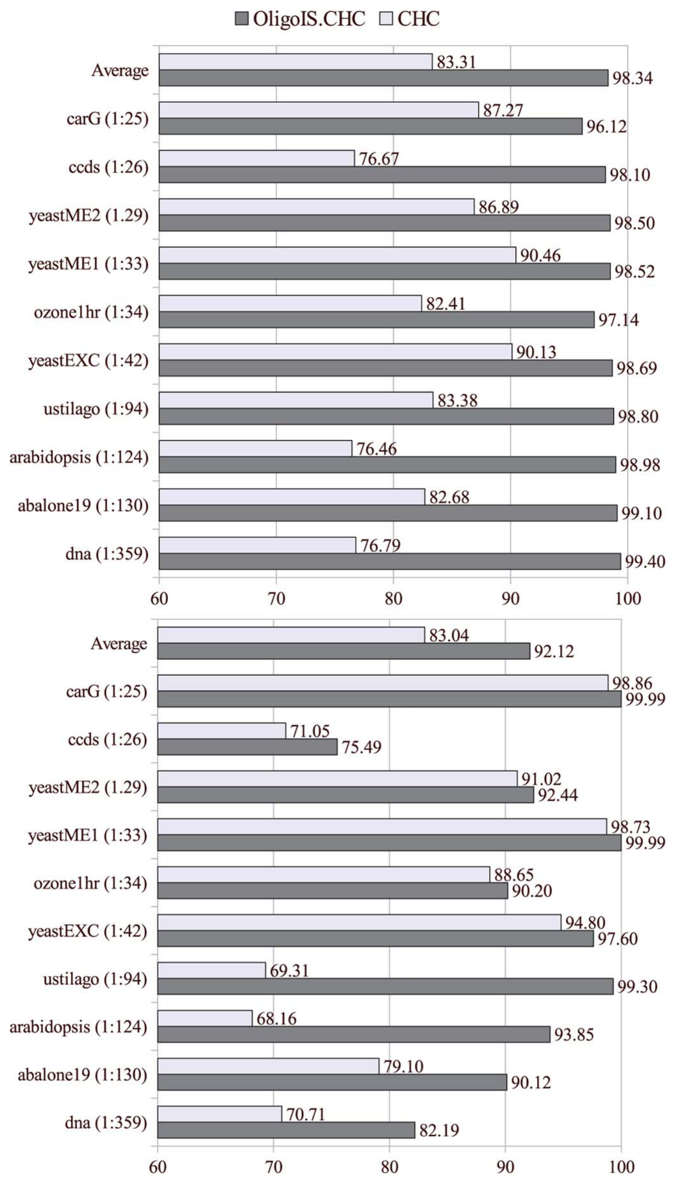

Fig. 6. (Top) Reduction and AUC (bottom) for OLIGOIS.CHC and CHC for the ten data sets with the highest IR.

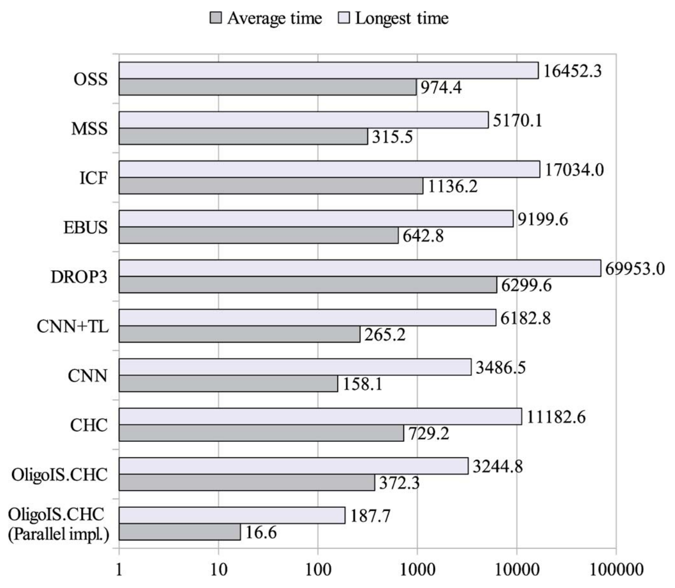

Fig. 7. Average time and longest time in logarithmic scale for the standard algorithms and OLIGOIS.CHC.

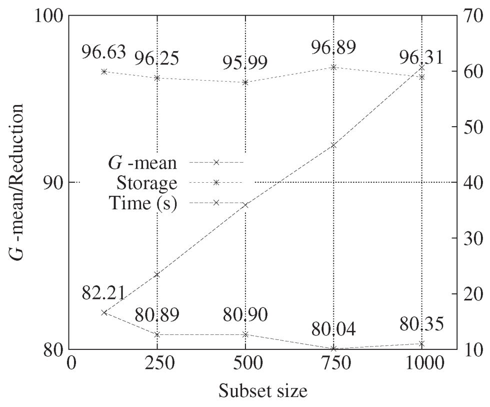

Fig. 8. Average G-mean, reduction, and execution time for OLIGOIS.CHC for different subset sizes.

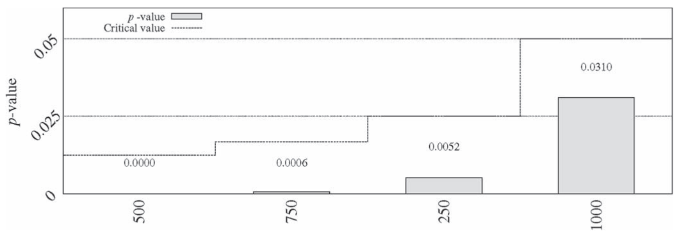

Fig. 9. Results of the Holm test for OLIGOIS.CHC reduction with a subset size of 100 instances as the control method, and OLIGOIS.CHC with subset sizes of 250, 500, 750 and 100 instances. Numerical p-values are shown.

For accuracy, the test obtained a p-value of 0.0998, and for reduction, the p-value was 0.6012, indicating that there were no significant differences in any of the cases. However, the results with ten rounds achieved the best average rank for both reduction and accuracy.

The last relevant parameter of our method is α [see (3)], which measures the relative importance of accuracy and reduction when evaluating the vote thresholds. We performed experiments with values of α of 0.3, 0.4, 0.5, 0.6, and 0.7. Fig. 11 shows the average accuracy, reduction, and execution time for these five values. As expected, lower values of α obtained higher reduction as the weight of the term in the evaluation of the thresholds of votes became more relevant. The price paid is lower accuracy.

The Iman–Davenport test obtained a p-value of 0 for reduction. This test also found significant differences for accuracy with a p-value of 0.0026. Therefore, we performed a Holm test for both accuracy and reduction. The results are shown in Fig. 12. We used $\alpha = 0.5$ as control experiments for accuracy because it achieved the best average rank for the G-mean value and $\alpha = 0.3$ for reduction for the same reason.

The test shows that decreasing α improves results in terms of reduction but harms accuracy. However, values above 0.5 do not show better accuracy than 0.5. Thus, this value is a good compromise between accuracy and reduction.

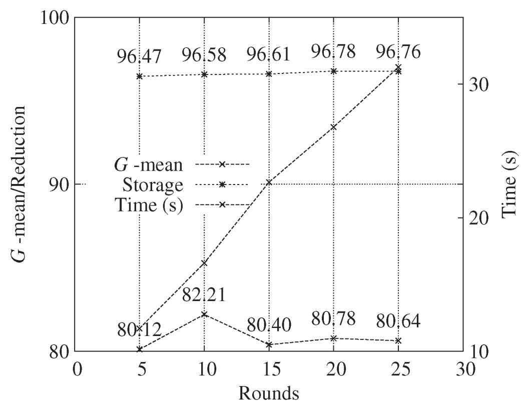

Fig. 10. Average G-mean, reduction, and execution time for OLIGOIS.CHC for different numbers of rounds.

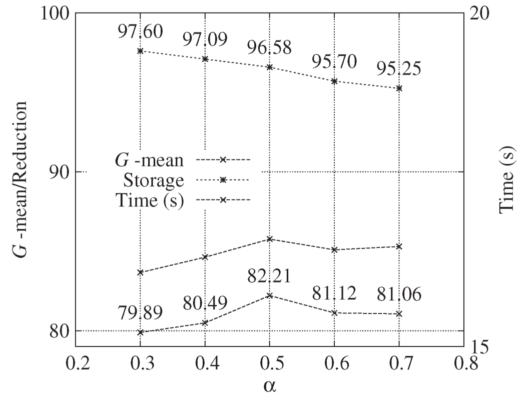

Fig. 11. Average G-mean, reduction, and execution time for OLIGOIS.CHC for different values of α.

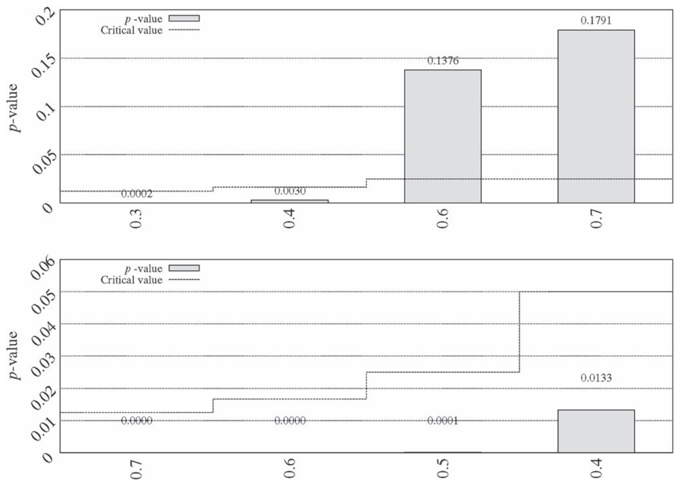

Fig. 12. Results of the Holm test for OLIGOIS.CHC as the control method with $\alpha = 0.5$ as the control experiment for (top) accuracy and with $\alpha = 0.3$ for (bottom) reduction, and OLIGOIS.CHC with the remaining values of α of 0.3, 0.4, 0.5, 0.6 and 0.7 for each case. Numerical p-values are shown.

C. Using Other Classifiers

As a final experiment, we tested whether OLIGOIS was able to improve results when using other classifiers. We repeated the experiments using the best version of our algorithm, i.e., OLIGOIS.CHC, and the best standard method, i.e., CHC. We used as classifiers a decision tree using C4.5 [31] algorithm and a support vector machine (SVM) [32] using a Gaussian kernel and the parameters γ and C obtained by tenfold cross-validation. Fig. 13 shows the results for these two classifiers. As was the case with 1-NN, OLIGOIS.CHC improves the results of CHC in both accuracy and reduction. Furthermore, the classifiers induced in both cases are simpler. All of the differences were found significant at a 0.05 significance level using a Wilcoxon test.

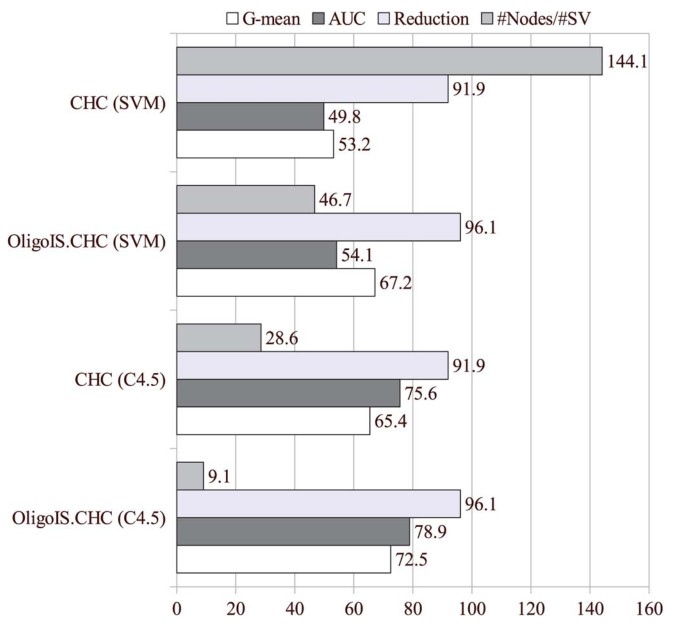

Fig. 13. Average G-mean, reduction, and classifier size for OLIGOIS.CHC and CHC using a C4.5 decision tree and a SVM.

TABLE VI
SUMMARY OF VERY LARGE DATA SETS

Dataset	Cases	Attributes			Inputs	IR	Memory
		C	B	N			size
ccds	364,495	-	-	1004	4016	1:25	5.4GB
chrom21	1,267,701	-	-	1003	4012	1:4913	18.9GB
dna	5,000,000	-	-	200	800	1:348	14.9GB

D. Scalability

In previous experiments, we showed the performance of OLIGOIS in data sets of medium size. However, one of the aims of this paper is the scalability of the presented method to very large data sets. Thus, we applied our algorithm to three very large problems. These data sets are shown in Table VI. `ccds` and `dna` represent the same problems as the corresponding data sets in Table I but with a larger set of instances. `chrom21` is the bioinformatics problem of predicting the translation initiation site of a gene in the sequence of human chromosome 21. With a largest set of five million instances and 800 features, these data sets represent a serious challenge for any method. Furthermore, `chrom21` and `dna` also show a large IR. To clarify the difficulty of the problem, we show the memory size of these three data sets using a standard float point precision stored with 4 B.

TABLE VII
SUMMARY OF RESULTS FOR LARGE DATA SETS. SIGNIFICANT DIFFERENCES FOR A RESAMPLED t-TEST ARE MARKED IN BOLDFACE

Dataset	Undersampling		OLIGOIS.CHC		
	G/AUC	Red.	G/AUC	Red.	Time (s)
ccds	67.02/0.7233	92.36	69.30/**0.7575**	**99.67**	1762.4
chrom21	79.12/0.8432	99.96	**81.71/0.8700**	99.90	3090.3
dna	72.04/0.7137	99.42	**75.93/0.7819**	99.39	5462.7

We applied OLIGOIS.CHC to a subset size of 1000 instances. As the control experiment, we used random undersampling because none of the other standard methods is scalable to data sets of this size. Table VII shows the results of both methods. The first interesting result is the good scalability of our proposal. For the largest data set of five million instances and 800 features, OLIGOIS.CHC took 1.5 h to complete the whole process. If we estimate the time that standard CHC would need for such a data set, considering that it needed more than 3 h for arabidopsis, which has 33 971 instances, we can conclude that OLIGOIS reduces the time by several orders of magnitude.

Furthermore, the performance of OLIGOIS.CHC is competitive. For ccds, there are no significant differences in terms of accuracy, but OLIGOIS.CHC achieves a significantly better reduction. For chrom21 and dna, both methods achieved the same reduction, but OLIGOIS.CHC obtained significantly better accuracy.

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented a new method of instance selection in class-imbalanced data sets that is applicable to very large data sets. The method consists of concurrently applying instance selection on small class-balanced subsets of the original data set and combining them by means of a voting method, setting a different threshold for minority and majority class samples. Using a CHC algorithm as the base algorithm, we have shown that our method exceeds the performance of state-of-the-art selection methods for class-imbalanced data in both accuracy and reduction. Its advantage over the state-of-the-art methods was larger when the IR of the problem was higher.

The results showed that our approach scaled up to problems of very large size. Two features of our method guarantee that scalability: First, instance selection is always performed over small data sets, keeping the time spent by the process low, and second, only small subsets of instances must be kept in memory, removing any scalability constraint due to memory limits. This scalability has been experimentally proven with the largest data set of five million instances and 800 features.

REFERENCES

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[2] N. V. Chawla, W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, Jan. 2002.

[3] C. Ling and G. Li, "Data mining for direct marketing problems and solutions," in *Proc. 4th Int. Conf. KDD*, New York, 1998, pp. 73–79.

[4] G. M. Weiss and F. Provost, "The effect of class distribution on classifier learning: An empirical study," Dept. Comput. Sci., Rutgers Univ., Newark, NJ, Tech. Rep. TR-43, 2001.

[5] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Comput. Intell.*, vol. 20, no. 1, pp. 18–36, Feb. 2004.

[6] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proc. 14th Int. Conf. Mach. Learn.*, 1997, pp. 179–186.

[7] S. García, J. Derrac, I. Triguero, C. J. Carmona, and F. Herrera, "Evolutionary-based selection of generalized instances for imbalanced classification," *Knowl.-Based Syst.*, vol. 25, no. 1, pp. 3–12, Feb. 2012.

[8] N. García-Pedrajas, J. A. Romero del Castillo, and D. Ortiz-Boyer, "A cooperative coevolutionary algorithm for instance selection for instance-based learning," *Mach. Learn.*, vol. 78, no. 3, pp. 381–420, Mar. 2010.

[9] S. García and F. Herrera, "Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy," *Evol. Comput.*, vol. 17, no. 3, pp. 275–306, Fall 2009.

[10] L. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, May 2003.

[11] N. García-Pedrajas, C. García-Osorio, and C. Fyfe, "Nonlinear boosting projections for ensemble construction," *J. Mach. Learn. Res.*, vol. 8, pp. 1–33, May 2007.

[12] R. Barandela, J. L. Sánchez, V. García, and E. Rangel, "Strategies for learning in class imbalance problems," *Pattern Recognit.*, vol. 36, no. 3, pp. 849–851, Mar. 2003.

[13] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[14] A. Frank and A. Asuncion, UCI Machine Learning Repository, 2010.

[15] N. García-Pedrajas, "Constructing ensembles of classifiers by means of weighted instance selection," *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 258–277, Feb. 2009.

[16] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 257–286, Mar. 2000.

[17] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972.

[18] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Mining Knowl. Discov.*, vol. 6, no. 2, pp. 153–172, Apr. 2002.

[19] L. Kuncheva, "Editing for the k-nearest neighbors rule by a genetic algorithm," *Pattern Recognit. Lett.*, vol. 16, no. 8, pp. 809–814, Aug. 1995.

[20] H. Ishibuchi and T. Nakashima, "Pattern and feature selection by genetic algorithms in nearest neighbor classification," *J. Adv. Comput. Intell. Intell. Inf.*, vol. 4, no. 2, pp. 138–145, 2000.

[21] C. R. Reeves and D. R. Bush, "Using genetic algorithms for training data selection in RBF networks," in *Instances Selection and Construction for Data Mining*, H. Liu and H. Motoda, Eds. Norwell, MA: Kluwer, 2001, pp. 339–356.

[22] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study," *IEEE Trans. Evol. Comput.*, vol. 7, no. 6, pp. 561–575, Dec. 2003.

[23] S. García, A. Fernández, and F. Herrera, "Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems," *Appl. Soft Comput.*, vol. 9, no. 4, pp. 1304–1314, Sep. 2009.

[24] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 3, pp. 515–516, May 1968.

[25] I. Tomek, "Two modifications of CNN," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. SMCB-6, no. 11, pp. 769–772, Nov. 1976.

[26] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 20–29, Jun. 2004.

[27] R. Barandela, F. J. Ferri, and J. S. Sánchez, "Decision boundary preserving prototype selection for nearest neighbor classification," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 19, no. 6, pp. 787–806, Sep. 2005.

[28] G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour, "An algorithm for selective nearest neighbor decision rule," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 6, pp. 665–669, Nov. 1975.

[29] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.

[30] J. Maudes-Raedo, J. J. Rodríguez-Díez, and C. García-Osorio, "Disturbing neighbors diversity for decision forest," in *Proc. Workshop SUEMA*, G. Valentini and O. Okun, Eds., Patras, Grecia, Jul. 2008, pp. 67–71.

[31] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[32] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1999.

Nicolás García-Pedrajas was born in Córdoba, Spain, in 1970. He received the B.S. degree in computing and the Ph.D. degree from the University of Málaga, Málaga, Spain, in 1993 and 2001, respectively.

He is currently a Professor in the area of computer science and artificial intelligence with the Department of Computing and Numerical Analysis, University of Córdoba, Córdoba, and the Head of the Computational Intelligence and Bioinformatics Research Group, University of Cordoba. His current research interests include classification, neural networks, evolutionary computation, and bioinformatics.

Javier Pérez-Rodríguez was born in Córdoba, Spain, in 1983. He received the B.S. degree in computing from the University of Córdoba, Córdoba, in 2008. He is currently working toward the Ph.D. degree in the University of Granada, Granada, Spain.

He is a member of the Computational Intelligence and Bioinformatics Research Group, University of Cordoba. His current research interests include evolutionary computation and bioinformatics.

Aida de Haro-García was born in Córdoba, Spain, in 1984. She received the B.S. degree in computing from the University of Córdoba, Córdoba, in 2007 and the Ph.D. degree from the University of Granada, Granada, Spain, in 2011.

She is currently an Assistant Professor in the area of computer science and artificial intelligence with the Department of Computing and Numerical Analysis, University of Córdoba. She is also a member of the Computational Intelligence and Bioinformatics Research Group, University of Córdoba. Her current research interests include classification, evolutionary computation, and feature and instance selection.