Pattern Recognition 46 (2013) 365–375

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

InstanceRank based on borders for instance selection

Pablo Hernandez-Leal [a,*], J. Ariel Carrasco-Ochoa [a], J.Fco. Martínez-Trinidad [a], J. Arturo Olvera-Lopez [b]

[a] National Institute of Astrophysics, Optics and Electronics, Computer Science Department, Luis Enrique Erro No. 1, Sta. María Tonantzintla, Puebla, CP 72840, Mexico
[b] Benemérita Universidad Autónoma de Puebla, Computer Science Department, Av. San Claudio y 14 Sur, Ciudad Universitaria, Puebla, CP 72570, Mexico

ARTICLE INFO

Article history:
Received 2 November 2011
Received in revised form
4 June 2012
Accepted 14 July 2012
Available online 31 July 2012

Keywords:
Instance selection
Instance ranking
Border instances
Supervised classification

ABSTRACT

Instance selection algorithms are used for reducing the number of training instances. However, most of them suffer from long runtimes which results in the incapability to be used with large datasets. In this work, we introduce an Instance Ranking per class using Borders (instances near to instances belonging to different classes), using this ranking we propose an instance selection algorithm (IRB). We evaluated the proposed algorithm using *k*-NN with small and large datasets, comparing it against state of the art instance selection algorithms. In our experiments, for large datasets IRB has the best compromise between time and accuracy. We also tested our algorithm using SVM, LWLR and C4.5 classifiers, in all cases the selection computed by our algorithm obtained the best accuracies in average.

1. Introduction

Supervised classification is a process in which, using a previously labeled set known as training set, individual instances are assigned to a class based on their characteristics. Supervised classification uses a training set (denoted in this document by T) for extracting (learning) models or rules in order to classify (assign a class or a label) unseen instances. The supervised classification task is carried out by algorithms known as classifiers. Commonly, an element $p \in T$ is named an instance, which has the form $p=(p_{a1},p_{a2},\ldots,p_{an})$ where p_{ai} is the i-th feature or attribute of p.

Supervised classification is used in many areas such as medicine [1–4], bioinformatics [5–7], economy [8,9] and industry [10]. In these areas commonly there are large datasets containing both categorical and numeric data. When classifiers extract models using T, it could happen that some instances in T do not provide relevant information in the learning phase, that is, there are irrelevant instances in T. The problem is to discard this kind of instances, therefore a process for discarding them must be applied. This process is known as Instance Selection (IS) [11–16], which selects a subset $S \subset T$ such that S does not contain irrelevant instances and Acc(S) is as high as possible, where Acc(S) is the classification accuracy obtained by using S as training for a classifier. Additionally, for large datasets, using a classifier with the whole training set sometimes would be unfeasible (given the time complexity), even with state of the art classifiers. An IS algorithm reduces the size of the training set and therefore training and classification runtimes are reduced too, mainly for instance-based classifiers (those that use the whole training set for classifying unseen instances). For these reasons, in order to obtain a subset of the best instances in T (that allows to obtain as high accuracy as possible) an IS algorithm is usually applied. Nevertheless, the majority of the IS algorithms fall into the same problem and they cannot deal with large datasets because their selection criteria need long runtimes. Therefore a fast IS algorithm is needed to perform instance selection in large datasets. In this work, a fast IS algorithm based on ranking is proposed. The main characteristic of our algorithm is that it constructs, taking into account border instances, a ranking for the instances of each class (which provides useful information for discriminating among classes). This ranking assigns high scores to border instances. Then, in order to have a good representation of each class, our algorithm selects some instances from three different ranges in the ranking: the best, the medium and the worst ranked. In our experiments, for small datasets the selection obtained by the proposed algorithm reaches an accuracy comparable to that obtained by the selection of DROP3 (one of the most successful IS algorithms); while for large datasets our algorithm has the best compromise between time and accuracy.

The rest of the paper is organized as follows: Section 2 presents works related to Instance Selection. Section 3 describes the proposed IS algorithm. Section 4 presents experiments for tuning the proposed IS algorithm parameters and for comparing its accuracy against state of the art IS algorithms. In Section 5 an evaluation in terms of accuracy and runtime is performed with large datasets. In Section 6, conclusions and future research directions are presented.

* Corresponding author. Tel.: +52 2663100.
E-mail address: pablohl@ccc.inaoep.mx (P. Hernandez-Leal).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.07.007

2. Related work

In the literature, there are different ways to categorize instance selection algorithms [13,17–19]. In this paper we will use the categorization proposed by [17,13] where IS algorithms are divided into two groups: Wrapper algorithms where the selection criterion is based on the accuracy obtained by a classifier, and Filter algorithms where the selection criterion is not based on a classifier. In the following subsections, some IS algorithms belonging to either wrapper or filter type are briefly described.

2.1. Wrapper

The selection rule in most wrapper algorithms is related to the k-NN (k-nearest neighbor) classifier [20]. An earlier IS algorithm is CNN (condensed nearest neighbor) [21], which starts with $S=\emptyset$ and randomly includes in S an instance from each class and repeatedly includes in S those instances in T misclassified by S. Another IS algorithm based on CNN is GCNN (generalized condensed nearest neighbor) rule [22] which includes in S instances of T, which are not represented by S. In the GCNN selection, an instance is represented when the difference between the distances to its nearest neighbors and its nearest enemies (i.e. nearest instances from a different class) are higher than a user specified threshold.

In practice, there exist noisy instances in the training sets, i.e. instances such that their description does not coincide with other instances belonging to the same class. ENN (edited nearest neighbor) [23] is an IS algorithm commonly used as noise filter. ENN deletes the noisy instances as follows: it discards an instance $p \in T$ when p does not coincide with the majority class of its k nearest neighbors, in particular, ENN uses $k=3$.

In [24], ICF (Iterative Case Filtering) algorithm was proposed, which selects instances based on the neighbors (Reachable set) and the associates (Coverage set) of an instance p. ICF deletes p whenever p has more neighbors than associates. For any instance p, the associate set is the set of instances having p as one of their k-nearest neighbors.

Other IS algorithms related to the associate set are the DROPs (Decremental Reduction Optimization Procedures) proposed in [25]. The DROPs discard an instance p if its associates can be correctly classified without p. According to the author's experimental results, the best DROP algorithms (among five proposed: DROP1… DROP5) are DROP3 and DROP5, which outperform other well known IS algorithms such as ENN and ICF.

The Class Boundary Preserving algorithm (CBP) presented in [26] also uses the concept of reachable set. They propose an extension of this concept using multiple levels, involving more than just the nearest enemy. The extension consists in using the i-th nearest enemy to obtain the reachable set of level i. Then, using the multiple reachable sets and the nearest enemies they determine geometric structure patterns in order to remove redundant instances.

The algorithm presented in [27] uses a clustering approach for instance selection that basically performs two steps. The first step groups instances of each class into clusters. The second step uses a wrapper process, which starts with an empty list of selected instances s, then it starts a cycle obtaining a random cluster c. From c the algorithm selects two random instances, one contained in s and other that is not contained in s. The algorithm swaps these instances and evaluates the accuracy; if this swap improves the accuracy, then s keeps the new instance. If not, the original instance is preserved. The cycle is repeated until either a maximum number of iterations is achieved or a stopping criterion is satisfied.

The IS algorithms described in the above paragraphs are strongly related to the use of the k-NN classifier but there exist IS algorithms that do not restrict the use of a specific classifier. Examples of this kind of algorithms are the evolutionary ones, which use the accuracy of a classifier as selection criterion. In these IS algorithms, an instance p is deleted whenever it does not contribute for either maintaining or improving the classification accuracy. An evolutionary IS algorithm is proposed in [28] where a memetic approach is used for selecting instances. The memetic algorithms combine evolutionary algorithms and local search, within the evolutionary cycle; a local search (among the chromosomes) is carried out in order to improve the accuracy and reducing the size of the solutions.

Other evolutionary IS algorithm is CCIS (Cooperative Coevolutionary Instance Selection) [29]. CCIS evolves two populations: the selector population (SP) and the combination population (CP). SP is obtained by splitting T into mutually exclusive blocks and crossing-mutating two blocks (selector subpopulations); CP is constructed via selecting an individual from each selector subpopulation. CCIS selects the best CP obtained after creating C generations of CP followed by P generations of SP.

A characteristic of the evolutionary IS algorithms (and also all those based on the classification accuracy as selection function) is the long runtimes required, since at each step, for deciding whether or not an instance should be removed from T, they invoke a classifier.

2.2. Filter

Some IS algorithms nonbased on a classifier have been proposed, this kind of algorithms are based on a selection function which is not based on a supervised classifier. In the following lines some works related to this approach are described.

The CLU (CLUstering) algorithm [30] is based on clustering. CLU divides T into c clusters and the selected instances are the centroids from each cluster. Another IS algorithm based on clustering is named PSC (Prototype Selection by Clustering) proposed in [31]. PSC selects border instances (instances near to class boundaries). In order to select border instances, PSC divides T into clusters and selects from nonhomogeneous clusters (which contain objects from more than one class) those instances nearest to another instance belonging to a different class. PSC not only selects border instances but also some nonborder ones (the centers of each homogeneous cluster) in order to retain representative instances in nonborder regions.

Another algorithm that selects border instances is presented in [32]. This work introduces a relation over pairs of instances, called class conditional nearest neighbor (CCNN). Using this relation two graphs are constructed, one for the information between instances in the same class (within-class graph) and another that represents the information for each pair of different classes (between-class graph). The IS algorithm has two steps: the first step removes "isolated" instances, for this, it uses a score based on the Kullback–Leibler divergence measure [33] using the degree distribution of the graphs, the instances with high score are retained. The second step is an iterative step that aims to remove instances selected in a previous step without affecting the accuracy.

A different way for selecting instances is via instance relevance, in this context, the most relevant instances in T are selected. The ISR (Instance Selection based on Ranking) [34] is an example of this type of algorithm inspired on the PageRank algorithm [35]. ISR computes the relevance of an instance within the training set through the typicality concept (the quotient of the instances average similarity with the rest of instances in its class and the average similarity with all those instances of a different class).

Once the relevance has been computed, ISR uses a wrapper process that includes in S (processing instances in a descending order according to their relevance) those instances yielding the highest number of correctly classified instances using T as test set. The selection process is repeated until either there are no misclassifications by S or there are no instances to process.

Another approach for reducing the number of instances is to produce a sparse representation of the dataset, this approach is used mainly by the area of sparse kernel selection methods [36,37]. The kernel methods use an elegant nonlinear generalization, with a kernel function implicitly transforming the feature space. The drawback with these methods is that the results are associated with every training instance. Thus, the idea is to reduce the number of examples, obtaining a sparse kernel representation of the data selecting the most appropriate samples that represent the kernel. However, these methods are still complex, in some cases in $O(n^3)$ [38], because of the optimization problem they have to solve.

Most of the IS algorithms discussed before have a selection criterion which needs long runtimes; this fact restricts the use of such algorithms, mainly for large datasets where a costly selection process is inapplicable. For this reason, some strategies for expediting the IS process have been proposed. Due their nature, these approaches do not constitute IS algorithms by themselves since they are fast ways for selecting instances using any IS algorithm. In [39] a strategy of this kind, which is based on the divide and conquer concept, is proposed. The main idea in the strategy consists in dividing the training set into small mutually exclusive blocks and then separately applying an IS algorithm to each block. The instances selected from each block are merged in subsets of approximately the same size and the IS algorithm is applied over these subsets. The process is repeated until a validation error starts to grow. Another strategy based on both classifier ensembles and the divide and conquer concept is proposed in [40]. Here, the idea consists of r rounds. In each round the training data is divided into disjoint blocks and any IS algorithm is separately applied over them. Those instances selected for removal in each round are marked and, using a voting criterion (like a classifier ensemble), the instances marked more than t times are removed from the training set. Given that these strategies accelerate any IS algorithm and the final runtime depends on the speed of the algorithm used for instance selection, the development of fast IS algorithms it is still a very important research area since the use of fast IS algorithms with these strategies would speed up even more the instance selection. In this paper, a fast IS algorithm is proposed. Our algorithm differs from others IS algorithms because it is based on computing a ranking for the instances of each class. Using this ranking, from each class different rank position instances are selected.

3. Proposed IS algorithm

In a training set, there exist either border or nonborder instances. Border instances are those near to decision boundaries between classes (instances near to instances belonging to different classes). On the other hand, nonborder instances are the opposite to the border ones.

After analyzing different IS algorithms we realized that many of them select border instances. For example, let us consider a bidimensional training set (Fig. 1(a)) and the results obtained by DROP3, DROP5 and PSC depicted in Fig. 1(b)–(d), respectively. According to these figures it can be noticed that the majority of the instances selected by them are border instances, despite these algorithms do not mention it. This gives us an intuition that these IS algorithms retain mainly border instances because those instances provide useful information for discriminating among classes.

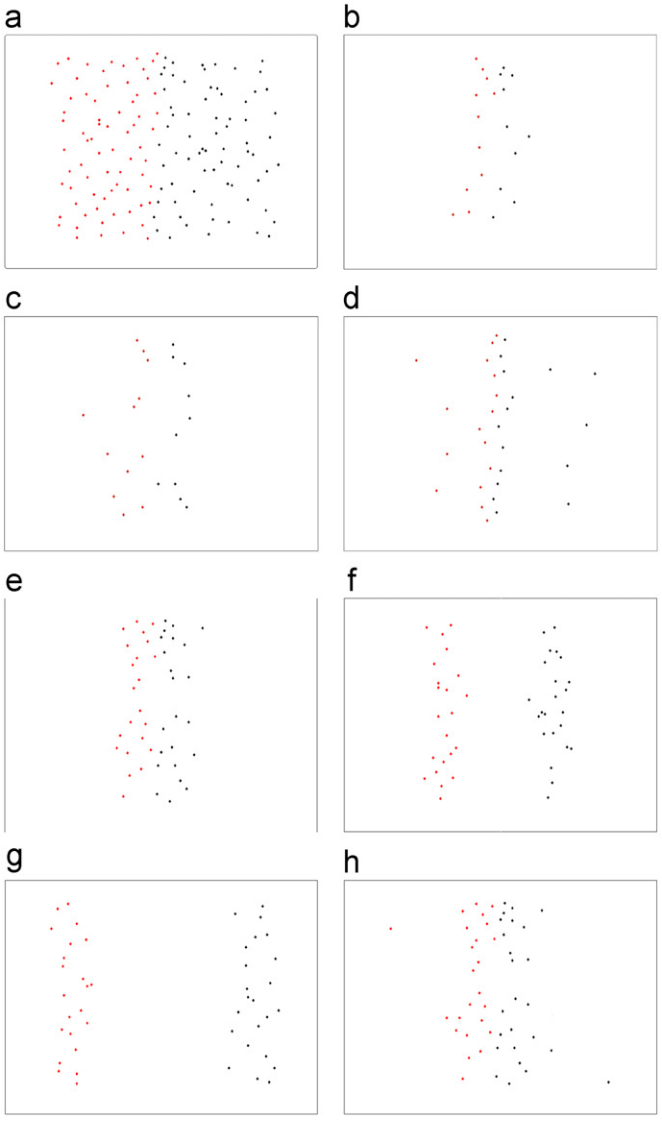

Fig. 1. Dataset and selected instances by different IS algorithms. (a) A dataset with two classes. (b) Instances selected by DROP3. (c) Instances selected by DROP5. (d) Instances selected by PSC. (e) 35% best ranked instances selected by IRB. (f) 35% medium ranked instances selected by IRB. (g) 35% worst ranked instances selected by IRB. (h) Instances selected by IRB (35,3,3).

Thus, the ranking idea proposed on this paper tries to preserve mainly borders of the dataset through the selection of border instances, that are those instances near to instances from other classes (nearest enemies).

Based on the idea of preserving border instances, we propose the IRB (InstanceRank based on Borders) algorithm, which computes the relevance of each instance of a class assigning high scores to border instances and in order to have a better representation of the dataset IRB selects some high (closer to the border), medium and low ranked (farthest to the border) instances. The IRB algorithm consists of three steps.

In the first step, like other IS algorithms [25], IRB uses the ENN algorithm to filter noise and to smooth borders, this is, to reduce the overlap between classes.

The second step involves ranking instances of each class, given an instance p_i belonging to the class C_r with $r=1,\ldots,m$; where m is the number of classes, IRB analyzes the nearest enemies of p_i

(nearest instances from a different class). The main idea behind the IRB algorithm is that the best ranked instances will be those near to instances belonging to different classes (enemies), i.e border instances. The IRB algorithm computes the rank of an instance p_i in a class C_r via the expression

$$rank(p_i \in C_r) = \frac{1}{k} \sum_{p_j \in E(p_i)} s(p_i,p_j) \quad (1)$$

where $s(p_i,p_j)$ is the similarity between p_i and p_j, $E(p_i)$ is the set of the k nearest enemies of p_i.

This expression computes the average of the similarities from an instance p_i to its k nearest enemies (p_j instances). This evaluation is applied to each instance of each class C_r. Therefore we obtain a ranking for the instances of each class in T.

Once the ranking has been computed, in the third step, IRB selects a percentage of instances from each class. In order to have a better representation of the dataset, IRB selects not only border instances, but also instances in other regions. For this reason, IRB selects border instances, the best ranked, and some nonborder instances with medium and low rank. The pseudocode of IRB is presented in Algorithm 1.

Algorithm 1. IRB pseudocode.

Input: *high*, *medium*, *low* percentage of instances to select from high-ranked, medium-ranked, and low-ranked instances respectively
T dataset
k number of nearest enemies used to compute the ranking
Result Instances selected T_{out}

```
T_enn = applyENNFilter(T);
foreach Class C do
    foreach instance p_i in T_enn in Class C do
        rank(p_i ∈ C) = (1/k) Σ_{p_j ∈ E(p_i)} s(p_i,p_j)
    end
end
sortInstancesByClass();
T_out = ∅
foreach Class C do
    T_out = T_out ∪ selectInstancesFrom(C,high,medium,low);
end
return T_out
```

The complexity analysis of IRB is straightforward. Since, in the worst case, each instance p_i could have, at most, $n-1$ enemies, the loop for obtaining the rank is $O(n^2)$, where n is the number of instances; sorting the instances is $O(n \log n)$; and the instance selection process is $O(n)$. Therefore IRB is $O(n^2)$.

In Fig. 1(e)–(h), the results of applying IRB on the dataset shown in Fig. 1(a) are presented. In Fig. 1(e) the best 35% ranked instances are depicted, in this figure we can observe that the best ranked instances tend to be closer to the border (border instances). In Fig. 1(f) the 35% of medium-ranked instances are depicted, in this figure the instances selected are farther from the border than the instances in Fig. 1(e). In Fig. 1(g) the 35% low-ranked instances are depicted, as we expected, these instances are the farthest from the border. Our algorithm takes into account this situation assigning the lowest values to farthest instances from the border. From these results we can observe that border instances are those with highest rank, therefore it would be desirable to select them. However, as it is shown in Fig. 1(b)–(d), IS algorithms also select some nonborder instances. For this reason IRB also selects some medium and low ranked instances. In Fig. 1(h) we show the selection made by IRB: 35% of the best ranked, 3% of the medium ranked, and 3% of the worst ranked. These three percentages (35,3,3) are parameters of our algorithm. An experimental study about these parameters is presented in Section 4.1.

4. Experiments over small and medium datasets

In this section we present three experiments. The first one has as objective studying the behavior of our algorithm when its parameters vary; through this study, we select the values used for the rest of the paper. The second experiment was performed to evaluate our algorithm using datasets commonly used in the literature for evaluating IS algorithms. Since most IS algorithms are not designed for large datasets, the datasets used in this experiment are small and medium size datasets. The third experiment evaluates the selection of the IS algorithms in terms of the accuracy they reach for different supervised classifiers, this is important because most of the IS algorithms only perform well using k-NN; however as we will show further IRB performs well with other classifiers.

For the three experiments shown in this section we used 10 small and medium datasets, which have been commonly used for evaluating IS algorithms (see Table 1). These datasets were taken from the UCI Repository [41]; we used five Numeric (Glass, Iris, Liver, Vehicle, Wine) and five Mixed (Echocardiogram, Heart-Cleveland, Heart-Swiss, Hepatitis, Zoo). For each experiment, the classification accuracy is obtained by the corresponding classifier using as training set the subset selected by each IS algorithm. In addition, the *Orig* column in each table indicates the accuracy obtained by using the original training set i.e. the whole training set without applying any instance selection algorithm. For computing instance relevance, our algorithm uses a similarity function. In this work, we used the HVDM (Heterogeneous Value Difference Metric) [42] function, since this function can be used with numeric and mixed datasets.

For the three experiments, we applied 10 fold cross validation over each dataset. In all tables presented in this section, the symbol "*" represents a statistically significant difference with respect to IRB according to the nonparametric Wilcoxon Signed test with a confidence level $\alpha = 0.05$ [43]. The results presented in *bold* represent the best accuracy scores for each dataset. All runtimes were obtained with a MacBook Pro with an Intel Core 2 Duo 2.4 GHz and 4 GB of RAM.

4.1. Parameters of our algorithm

We perform a series of experiments in order to obtain the best values for the parameters of the proposed IS algorithm (% of high, medium and low ranked instances). The first experiment was done to evaluate the accuracy reached by the selection obtained by IRB while increasing the percentage of border instances (high ranked). We repeated the experiment 10 times using 10 fold cross

Table 1
Number of instances, number of numeric attributes (Attr-Num) and number of categorial attributes (Attr-Cat) of the datasets used.

Dataset	Instances	Attr-Num	Attr-Cat
Echocardiogram	132	7	2
Glass	214	10	0
Heart Cleveland	303	5	8
Heart Swiss	123	5	8
Hepatitis	155	6	13
Iris	150	4	0
Liver	345	7	0
Vehicle	846	18	0
Wine	178	13	0
Zoo	101	0	16

validation for each value of the first parameter from {20, 25, 30, 35, 40}. The results of these experiments are presented in Table 2. The row "difference" represents the difference between the average of a column and the following one on the left. From this table, we can notice that the accuracy increases if the first parameter also increases. However, the improvement in accuracy from increasing the first parameter decreases each time. From these experiments we decided to select 35% as the default value for the first parameter for the following experiments, since the improvement of using 40% instead of 35% is very small (0.02). The second experiment evaluates separately each ranking level (high, medium and low). Therefore we tested IRB with (35,0,0) (0,35,0), (0,0,35), the results of these experiments are presented in Table 3. We can see that the best results were obtained with (35,0,0), which produced the best accuracy in average and the best accuracy for nine from 10 datasets. This was a significant result because it gives the intuition that our ranking performs well, i.e. border instances (instances with high rank) are the most important. Next, we decided to increase the last two parameters (medium and low), this was made with the idea of getting a better representation of the dataset by including some nonborder instances. The results are presented in Table 4. The row "difference" represents the difference between the average of a column and the following one on the right. In Table 4 we can observe that, if we increase the last two parameters, the accuracy in average increases. However, this difference is zero from (35,3,3) to (35,4,4). Based on these results, we decided to use (35,3,3) for the rest of the experiments. These results show that border instances are the most important to be retained but also some nonborder instances must be retained.

4.2. Comparison against other instance selection algorithms

In this section, we report a comparison among IRB(35,3,3) (the parameter values selected for our algorithm in the previous experiments), ISR, DROP3, DROP5, CLU and PSC instance selection algorithms. IRB was programmed in Java. For ISR we used the program written by the authors in Java. For the DROPS we used the C program written by the authors. For PSC we used the Matlab implementation written by the authors. For CLU we implemented the algorithm in Matlab. We have considered these algorithms in our experiments because according to the results reported in [25,24] the DROPs outperform in accuracy other relevant algorithms such as ENN and ICF. ISR was included since it is an IS algorithm based on ranking and CLU and PSC were included since they are two of the fastest instance selection algorithms. The algorithms GCNN, CCNN, CBP were not included because according to results reported in [22,32,26] they are competitive in accuracy against DROP3 and their runtimes are similar or longer than DROP3's.

It is important to highlight that we did not compare against strategies designed to accelerate the IS process as [39,29,40] because they are not IS algorithms, they are strategies that can be applied to speed up any IS algorithm. Moreover, since these strategies are based on the divide and conquer approach, the reduction in time would be of the same order for any IS algorithm.

For selecting the values for the parameters of our algorithm we followed the same methodology used for CLU [30] and PSC [31]. For these algorithms it is required to fix the number of clusters N to be created from the training set. In [31] several values of N were tested and the best ones were $N=8C$ and $N=6C$ for CLU and

Table 3
Classification accuracy results obtained by testing IRB with different parameters using k-NN ($k=3$).

Dataset	IRB(35,0,0)	IRB(0,35,0)	IRB(0,0,35)
Echocardiogram	**91.76**	88.11	86.22
Glass	**64.44**	59.30	56.73
Heart Cleveland	**82.87**	81.02	81.45
Heart Swiss	**93.50**	92.60	**93.50**
Hepatitis	**83.42**	82.65	80.77
Iris	**93.26**	92.93	89.13
Liver	**62.35**	59.57	53.04
Vehicle	66.43	63.62	**66.52**
Wine	**84.66**	81.91	84.22
Zoo	**85.10**	81.20	82.10
Average	**80.78**	78.29	77.57

Table 4
Classification accuracy results obtained using k-NN ($k=3$).

Dataset	35,4,4	**IRB(35,3,3)**	35,2,2	35,1,1	35,0,0
Echocardiogram	**93.39**	**93.39**	**93.39**	**93.39**	**93.39**
Glass	**63.98**	**63.98**	62.75	61.65	61.65
Heart Cleveland	**81.80**	**81.80**	**81.80**	81.32	81.32
Heart Swiss	**93.46**	**93.46**	**93.46**	**93.46**	**93.46**
Hepatitis	84.54	**85.17**	84.54	84.54	84.54
Iris	**94.67**	**94.67**	**94.67**	**94.67**	**94.67**
Liver	63.18	63.46	63.75	**65.19**	64.03
Vehicle	**65.36**	65.01	64.77	64.89	62.41
Wine	**89.28**	88.73	88.73	88.73	88.17
Zoo	**90.00**	**90.00**	**90.00**	**90.00**	**90.00**
Average	**81.97**	**81.97**	81.79	81.78	81.36
Difference	0.00	0.18	0.01	0.42	–

Table 2
Classification accuracy results obtained by testing IRB with different values for the parameter *high* that represent border instances using k-NN ($k=3$).

Dataset	IRB(20,0,0)	IRB(25,0,0)	IRB(30,0,0)	IRB(35,0,0)	IRB(40,0,0)
Echocardiogram	92.01	**92.59**	91.75	90.57	90.79
Glass	59.40	60.69	59.97	**60.74**	61.13
Heart Cleveland	79.21	79.73	79.78	**80.52**	77.56
Heart Swiss	**93.53**	**93.53**	**93.53**	**93.53**	92.09
Hepatitis	83.34	82.98	83.87	84.83	**84.99**
Iris	91.33	92.53	93.47	**93.93**	93.74
Liver	59.80	61.92	62.87	60.67	**64.73**
Vehicle	57.83	58.83	61.79	62.74	**64.64**
Wine	93.16	93.21	**94.32**	93.92	90.64
Zoo	83.84	85.05	85.45	**89.69**	91.03
Average	79.34	80.10	80.68	81.11	81.13
Difference	–	0.76	0.58	0.43	0.02

PSC respectively, where C is the number of classes in the training set. In our experiments, we use these values for N.

For DROPs and ISR we used the best values for their parameters, which were suggested by their authors [34,25]. ISR uses a specific classifier that is a variant of k-NN ($k=1$) using weights, the results of this algorithm are shown in the column "ISR". Even more, we also present the results for the ISR algorithm with the standard k-NN ($k=3$) classifier, these results are presented in the column "ISR-k-NN".

In Table 5 we present the results of applying k-NN ($k=3$) using as training the subsets selected by each evaluated IS algorithm. In these experiments the best results were obtained by IRB followed by ISR (with its special classifier). When ISR is used with the standard k-NN, the accuracies decrease approximately 15%. The DROP algorithms obtained very similar accuracy results but lower than ISR. In these experiments CLU obtained the lowest accuracies.

Even though IRB obtained the best accuracy it is important to notice that, in most of the results, there is no significant statistical difference with the other IS algorithms.

4.3. Comparison using other classifiers

In contrast with most IS algorithms which only perform well using k-NN. One of the most interesting advantages of our IS algorithm is that it performs well with other classifiers.

In order to show this characteristic, we tested the subsets selected by each algorithm as training for LWLR, SVM and C4.5 classifiers. These classifiers were chosen because they represent three quite different approaches for supervised classification. For these classifiers we used their Weka implementations with the default parameters [44]. We report the results in Tables 6–8.

For these three classifiers IRB was the best IS algorithm in average. For the LWLR classifier, PSC obtained the second best accuracy in average. The DROPs and ISR obtained lower accuracy results than PSC. The worst result was obtained with CLU. For SVM, PSC and DROPs algorithms obtained similar accuracy, but lower than IRB, CLU obtained the worst accuracy. For C4.5, the algorithm PSC obtained the second best accuracy in average. ISR obtained the worst results and in most of the datasets it has a statistical significant difference with IRB.

Taking into account the three classifier results, IRB obtained the best results in average.

5. Experiments over large datasets

In this section we present four different experiments. The first one evaluates runtimes and accuracy results while increasing the size of the datasets. The second experiment was performed to evaluate the selection with other classifiers (SVM, C4.5, LWLR). The third experiment shows the scalability of our algorithm with large datasets. Given that the datasets used in these experiments

Table 6
Classification accuracy for the LWLR classifier.

Dataset	Orig	IRB	ISR	DROP3	DROP5	CLU	PSC
Echocardiogram	95.71	93.04	86.07*	90.35	88.74	93.03	**97.50**
Glass	57.85	53.68	47.61	50.09	52.38	46.70*	**56.06**
Heart Cleveland	71.93	72.62	72.48	65.35*	73.18	**75.18**	69.93
Heart Swiss	96.42	**93.72**	93.46	**93.72**	92.88	92.04	76.72*
Hepatitis	79.99	**82.58**	68.54*	64.58*	70.29*	74.16*	73.37*
Iris	98.00	94.67	88.66*	92.00	92.00	88.00*	**96.00**
Liver	70.13	57.46	52.74	68.26*	**69.54***	48.39*	68.57*
Vehicle	44.90	44.90	45.64	**49.90**	48.30	38.4*	40.30
Wine	92.15	86.54	76.30*	62.75*	60.78*	53.88*	**88.72**
Zoo	83.32	77.78	87.77*	**88.88***	77.77	82.21	88.54*
Average	79.04	**75.69**	71.93	72.58	72.58	69.19	75.57

Table 7
Classification accuracy for the SVM classifier.

Dataset	Orig	IRB	ISR	DROP3	DROP5	CLU	PSC
Echocardiogram	93.21	**93.21**	71.96*	**93.21**	88.92	**93.21**	89.28
Glass	72.29	44.05	45.32	59.74	63.50*	56.03*	**65.84***
Heart Cleveland	84.79	82.10	76.70	81.18	**82.12**	80.21	81.17
Heart Swiss	92.25	**93.80**	93.46	93.71	93.71	92.05	85.44
Hepatitis	86.49	**83.50**	79.83	69.71*	69.71*	77.46	73.45*
Iris	96.66	**94.67**	84.00*	91.33	93.33	85.33*	93.33
Liver	70.72	57.98	56.44	58.26	56.78	52.50*	**64.07**
Vehicle	74.10	60.00	56.03	54.00	**65.90**	26.3*	44.30*
Wine	97.18	93.20	89.83	94.93	92.71	74.21*	**95.52**
Zoo	95.55	87.78	92.22	88.88	83.33	**95.55***	**95.55***
Average	86.32	**79.02**	74.58	78.49	79.00	73.28	78.79

Table 8
Classification accuracy for the C4.5 classifier.

Dataset	Orig	IRB	ISR	DROP3	DROP5	CLU	PSC
Echocardiogram	95.71	93.21	70.35*	84.10	92.85	**94.46**	93.10
Glass	67.29	59.74	51.79	60.19	53.76	49.67	**60.58**
Heart Cleveland	71.96	74.85	65.93*	68.59	72.16	**75.58**	69.89
Heart Swiss	93.71	**93.71**	93.46	**93.71**	**93.71**	92.05	87.05
Hepatitis	76.70	**82.50**	72.75*	63.33*	63.41*	72.79*	73.50
Iris	93.99	**94.00**	58.00*	92.66	90.66	82.66	90.66
Liver	63.67	**64.64**	51.84*	59.48	63.67	54.19	63.67
Vehicle	73.80	66.00	49.30*	57.40	73.60	38.8*	**74.00**
Wine	94.44	**91.50**	63.59*	84.43	78.88	75.55*	90.77
Zoo	93.33	77.78	42.22*	81.10	88.88*	91.11*	**93.33***
Average	82.46	**79.79**	61.92	74.49	77.15	72.68	79.65

Table 5
Classification accuracy for the selected algorithms, using k-NN ($k=3$).

Dataset	Orig.	IRB	ISR-k-NN	ISR	DROP3	DROP5	CLU	PSC
Echocardiogram	94.82	93.39	69.93	91.96	**94.29**	91.61	79.80	75.50*
Glass	72.94	63.98	53.70	62.16	**64.03**	62.71	54.20*	59.43
Heart Cleveland	83.70	81.80	**84.54**	81.77	66.02*	63.10*	72.60*	65.6*
Heart Swiss	91.02	**93.46**	93.40	93.40	93.40	93.40	93.40	78.84*
Hepatitis	79.95	**85.17**	78.66	76.79*	79.30	76.00	76.60	79.37
Iris	94.66	**94.67**	58.00	94.00	92.67	**94.67**	89.33	94.66
Liver	61.76	63.46	56.72	60.81	**64.90**	64.00	52.70*	59.30
Vehicle	70.69	**65.01**	52.24	57.80*	59.90	60.40	41.61*	62.07
Wine	92.7	88.73	74.70	**96.07**	94.40	93.86	87.37	92.67
Zoo	95.55	90.00	38.88	93.33	90.00	**95.56**	93.33	93.33
Average	83.78	**81.97**	65.98	80.80	79.89	79.53	74.10	76.07

are large, the subsets obtained by IRB are also large, therefore the last experiment in this section shows the effect of decreasing the values of the parameters of IRB for large datasets.

5.1. Comparing against other IS algorithms

In order to show that IRB is a fast IS algorithm we performed a series of experiments with datasets larger than those used in the previous section. We compared our approach with CLU and PSC because they are fast IS algorithms, we also included DROP3 because it is an algorithm that produces high accuracy. ISR was not included in these experiments due to its high space requirements for obtaining their ranking, therefore applying ISR is unfeasible for large datasets. For these experiments we also used 10 fold cross validation, as well as the Wilcoxon signed test with a confidence level $\alpha = 0.05$ [43] for evaluating statistically significant differences with IRB, which are marked with an "*" in the tables.

In Table 9 we show the time spent (in seconds) by each IS algorithm for selecting instances from 11 datasets, five with categorical data (Chess, Poker 90k, Poker 350k, Covertype, Census Income KDD) and six Numeric datasets (Text, Segmentation, Magic, Letter, USPS, Shuttle). All these datasets were obtained from the UCI Repository [41], except USPS obtained from [45], and obtained from [46]. The Text dataset is a collection of 375 documents in Spanish about natural disasters in Mexico, divided into four classes: hurricane, inundation, drought, and irrelevant. This dataset has 2550 attributes, which represent the terms (words) with frequency >10. Additionally, in Table 9, we show the number of instances and the number of attributes of each dataset. The "–" sign indicates that the experiment took more than 100 h (360,000 s) for only one fold. The "Total" row shows the sum of runtimes where all IS algorithms could be applied (Text, Segmentation, USPS, Magic, Letter, Chess, Shuttle and Poker 90k). The results show that DROP3 is very slow when the dataset has more than 30,000 instances. On the other hand CLU and PSC are IS algorithms that run very fast with numeric datasets but they do not run too fast with mixed datasets. The proposed algorithm was the fastest for the largest datasets with mixed data, and their runtimes were much shorter than the runtimes of DROP3 for large numeric datasets.

In Table 10 we present the accuracy (using k-NN, $k=3$) obtained by the different IS algorithms for the large datasets shown in Table 9. The average was computed taking into account only those datasets where all IS algorithms could be applied in less than 100 h per fold (first seven datasets of Table 9). For those datasets where all the algorithms finished their executions, DROP3 obtained the best accuracy in average, but for the four largest datasets it did not produce a result after 300 h. PSC obtained the third best accuracy but it is far from the best accuracy. PSC runtimes dramatically grows for mixed datasets. The same occurs with CLU which was the algorithm with the worst accuracy results. This behavior for CLU and PSC on mixed datasets is explained due to the fact that computing the centroid in a cluster for mixed data is much more complex.

Table 10
Classification accuracy results obtained by different IS algorithms for large datasets using k-NN ($k=3$).

Dataset	Orig	IRB	Drop3	CLU	PSC
Text	90.13	87.20	88.01	73.60*	**88.59**
Segmentation	95.19	89.48	91.24*	79.10*	**91.43**
Magic	83.80	**81.14**	80.89	67.75*	72.86*
Letter	94.90	88.02	**92.47***	42.52*	77.89*
Chess	56.70	51.31	**55.06***	30.20*	43.36
USPS	95.46	91.59	**94.76***	78.25*	91.44
Shuttle	99.89	**99.79**	99.75	95.00*	96.80*
Poker 90k	61.54	54.90	–	–	–
Covertype 250k	92.73	86.50	–	–	–
Census Income KDD	93.91	94.70	–	–	–
Poker 350k	63.84	53.10	–	–	–
Average	88.01	84.07	**86.03**	68.63	80.34

Table 11
Classification accuracy for the LWLR classifier.

Dataset	Orig	IRB	DROP3	CLU	PSC
Text	90.05	**88.35**	66.95*	83.10*	87.53
Segmentation	77.43	**73.14**	39.81*	19.48*	37.62*
Magic	72.38	**74.14**	77.33*	54.71*	66.46*
Letter	40.27	24.47	**28.46***	6.36*	9.73*
Chess	29.44	33.60	**35.50**	13.90*	34.68
USPS	33.86	32.15	**33.83**	15.98*	20.94*
Shuttle	86.89	**86.90**	76.12	69.86*	75.48*
Average	61.47	**58.96**	51.00	37.63	47.49

Table 12
Classification accuracy for the SVM classifier.

Dataset	Orig	IRB	DROP3	CLU	PSC
Text	92.03	**88.87**	83.54	77.93*	91.47
Segmentation	92.86	88.86	**90.76***	21.57*	82.76*
Magic	79.16	**78.49**	76.83*	67.59*	68.35*
Letter	82.30	**76.55**	75.28*	6.37*	67.19*
Chess	47.76	45.95	**46.11**	11.22*	42.12
USPS	95.22	92.87	**93.98***	18.70*	94.58*
Shuttle	96.96	**95.55**	91.39*	79.37*	78.83*
Average	83.76	**81.02**	79.67	40.39	75.04

Table 9
Runtimes (sec) for the large datasets for different IS algorithms using k-NN ($k=3$).

Dataset	#Instances	Attr-Num	Attr-Cat	IRB	CLU	PSC	DROP3
Text	375	2550	0	40.5	**4.9**	6.0	21.1
Segmentation	2100	19	0	27.0	**6.0**	7.0	53.0
USPS	9298	256	0	894.6	**432.4**	448.8	11,677.5
Magic	19,020	10	0	211.5	**167.1**	172.1	2555.6
Letter	20,000	16	0	500.8	**217.2**	226.2	4765.5
Chess	28,056	0	6	**1651.4**	3762.3	3862.8	7447.2
Shuttle	58,000	9	0	2470.7	**277.4**	288.4	123,000.1
Poker 90k	90,000	0	10	**20,301.4**	58,813.1	60,519.5	68,834.1
Covertype 250k	250,000	10	44	**150,139.8**	–	–	–
Census Income KDD	299,285	7	33	**230,375.7**	–	–	–
Poker 350k	350,000	0	10	**313,667.1**	–	–	–
Total				**26,097.9**	63,680.4	65,530.8	218,354.1

CLU and PSC compute the centroid as the instance most similar to the remaining instances in a cluster. In contrast, the centroid for numeric data is computed as the average of the instances in a cluster. These two algorithms use a clustering algorithm that needs to obtain centroids, therefore the runtimes increase very fast. Finally, IRB obtained the second best results in accuracy and it was the only algorithm able to be applied on all the tested datasets.

5.2. Comparison using other classifiers

We also performed experiments using the selected instances as training for other classifiers (LWLR, SVM and C4.5). In this experiment we only use those datasets where all IS algorithms could be applied (see Section 5.1). The results are presented in Tables 11–13. For all the tested classifiers, in average, our proposed algorithm, IRB, obtained the best accuracy results.

Table 13
Classification accuracy for the C4.5 classifier.

Dataset	Orig	IRB	DROP3	CLU	PSC
Text	90.87	**89.01**	86.89	86.86	85.11
Segmentation	96.14	**90.90**	83.57*	19.76*	89.10
Magic	84.93	**83.42**	82.97	61.08*	82.42*
Letter	88.12	**76.91**	74.26*	6.88*	72.51*
Chess	62.02	46.53	46.61	10.88*	**48.49***
USPS	88.26	**83.23**	74.92*	16.47*	79.61*
Shuttle	99.97	**99.89**	99.86	61.11*	99.10
Average	87.19	**81.41**	78.44	37.58	79.48

Table 14
Runtimes for the different IS algorithms for the Covertype dataset.

# Instances	IRB	DROP3	PSC	CLU
10,000	151.6	1336.0	853.4	833.2
20,000	626.0	7648.9	2709.7	2668.4
30,000	1421.9	27,600.4	6100.9	6132.3
40,000	2424.1	–	25,560.2	25,231.9
50,000	4448.6	–	–	–
60,000	6460.4	–	–	–
70,000	8804.3	–	–	–
80,000	11,170.6	–	–	–
90,000	13,710.0	–	–	–
100,000	15,650.1	–	–	–

DROP3 obtained lower accuracy than using k-NN as we expected since DROP3 was designed for the k-NN classifier. CLU obtained the worst accuracy in average for all the classifiers.

5.3. Scalability

In this section we used one large dataset to show the scalability of the IS algorithms. The experiment was performed with the Covertype dataset (54 features, seven classes, 250,000 instances, mixed data). This dataset was selected because it was one of the largest datasets tested in Section 5.1. For this dataset, in order to show the scalability of the algorithms, we constructed 10 training sets from 10,000 instances to 100,000 instances.

The runtimes results for the Covertype dataset are presented in Table 14 and Fig. 2. The results show that DROP3 runtimes grew very fast. For PSC and CLU their runtime growth tendency was similar to DROP3. This happens because this dataset is not numeric; therefore CLU and PSC increase their runtimes, because clustering this kind of data is more expensive in time. The proposed algorithm IRB was clearly the fastest IS algorithm.

The accuracy results for the Covertype dataset, using k-NN ($k=3$), are shown in Table 15. For the Covertype dataset PSC and CLU obtained low accuracy. DROP3 obtained the best accuracy for 10,000, 20,000 and 30,000 but its runtimes grew very fast for large datasets as it can be seen in Fig. 2. IRB was the fastest algorithm with the second best accuracy.

5.4. Retention for large datasets

In all the previous experiments we used the parameter values (35,3,3) for IRB since these values obtained the best results for

Table 15
Classification accuracy for different IS algorithms for the Covertype dataset.

Instances	IRB	DROP3	PSC	CLU
10,000	65.68	71.71	50.75	35.88
20,000	68.13	73.14	49.63	38.94
30,000	69.34	74.32	61.42	49.82
40,000	70.09	–	62.50	53.40
50,000	71.32	–	–	–
60,000	71.78	–	–	–
70,000	73.06	–	–	–
80,000	73.51	–	–	–
90,000	74.20	–	–	–
100,000	74.27	–	–	–

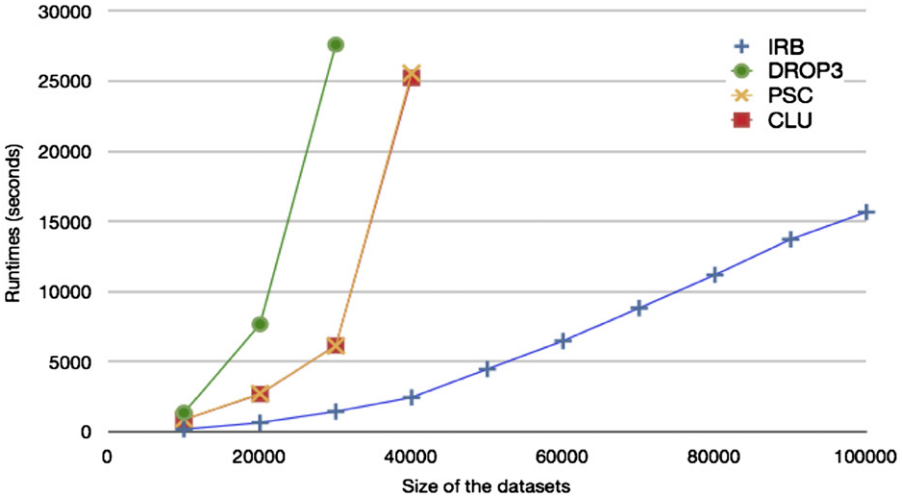

Fig. 2. Runtimes spent by each algorithm over different training sets size from the Covertype dataset.

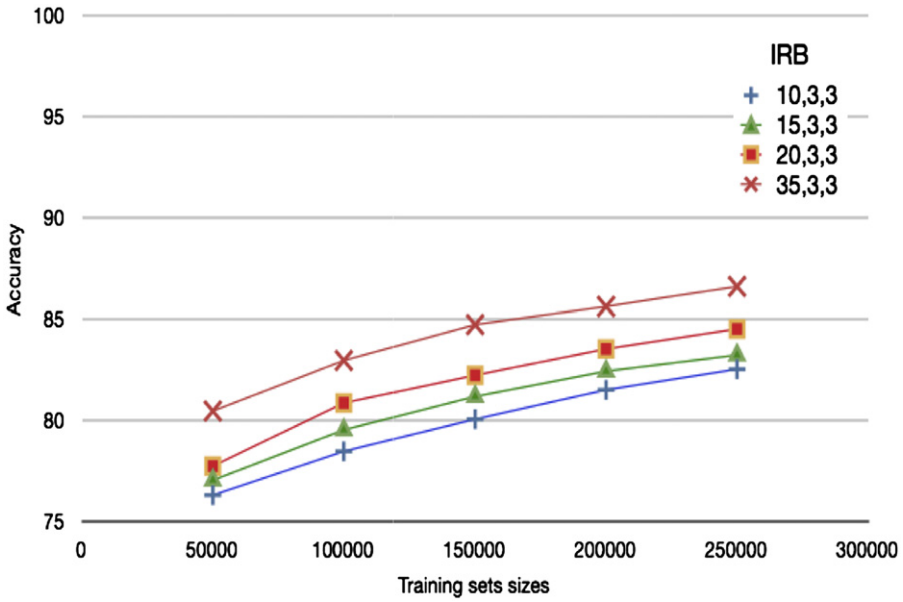

Fig. 3. Accuracies obtained with three different sets of parameters using IRB for the Covertype dataset.

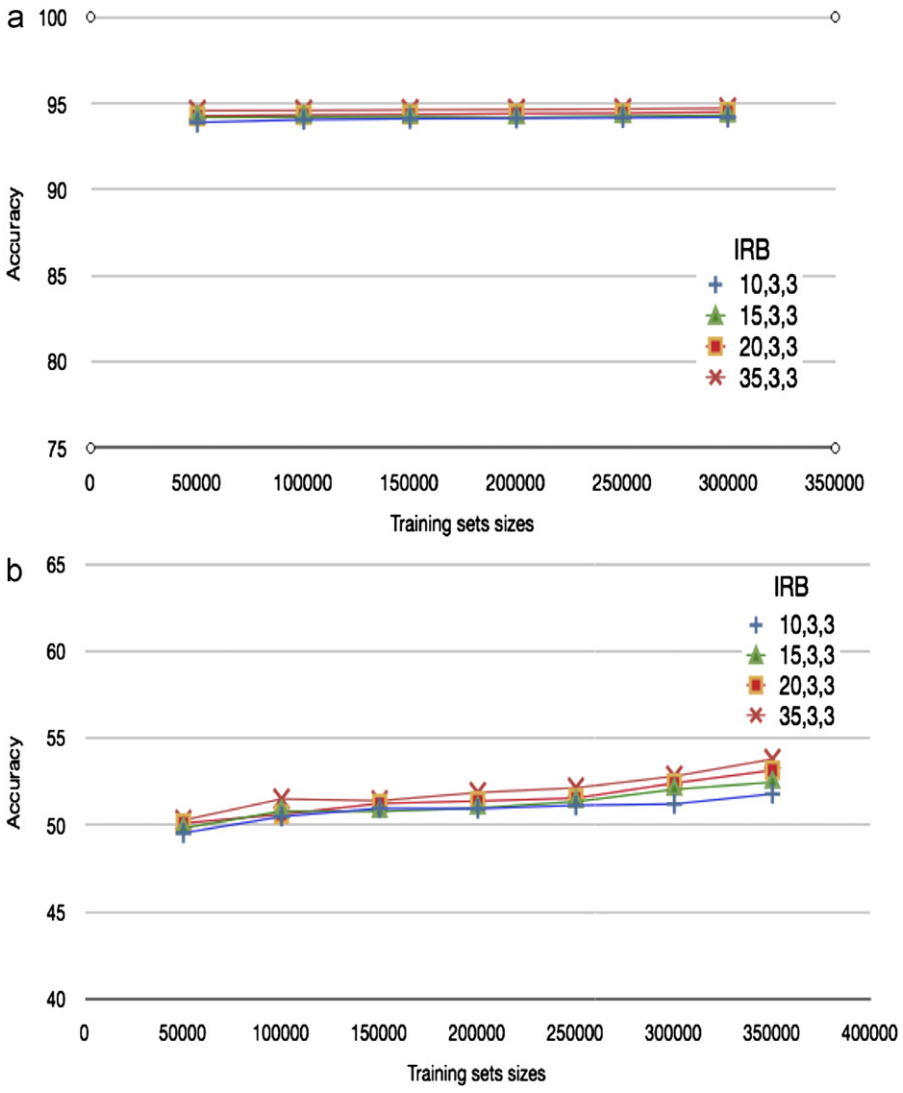

Fig. 4. Accuracies obtained with three different sets of parameters using IRB for two datasets. (a) Census, (b) Poker 350k.

small and medium size datasets as it was explained in Section 4.1. For these values of parameters the retention of the algorithm was approximately 40%. However, for some domains, specially for large datasets, it would be needed to obtain a reduced number of instances, for this reason we present three experiments to show how reducing the parameter values of IRB affects the accuracy of our algorithm for large datasets where other algorithms cannot be applied. It is important to note that the change in the parameters will not increase the runtimes because the ranking is obtained in the same way, only the number of selected instances will change.

In Figs. 3 and 4(a) and (b) we present the accuracy results using three fold cross validation using the *k*-NN classifier for the three largest datasets presented in Table 9. For each dataset we constructed different training sets starting in 50,000 instances and increasing them in the same size. For each training set we applied IRB with four different sets of parameters (35,3,3), (20,3,3), (15,3,3) and (10,3,3).

We can observe that for large datasets IRB obtains the best results with (35,3,3), but the difference with the other parameter values is not too big, therefore since our algorithm does not have competitor for large datasets we could reduce the values of IRB's parameters in order to select a set of instances around 15% of the training set size.

6. Conclusions

In this work we introduced the InstanceRank based on Borders (IRB) algorithm. Our algorithm constructs a ranking per class and selects some instances according to this ranking. We evaluated the proposed algorithm with small and large datasets. For small datasets its accuracy was comparable to DROP3, one of the most successful IS algorithms.

For large mixed datasets IRB is the fastest IS algorithm. For numeric datasets IRB is not the fastest algorithm but it obtains the best accuracy in a reasonable time. We also tested our algorithm with other classifiers (SVM, LWLR and C4.5) from quite different approaches. In all cases our algorithm obtained the best accuracies in average. Finally we present experiments showing that for large datasets IRB can select about the 15% of instances, changing the values of its parameters, with a small effect in the accuracy.

As future work, we propose to study other strategies for ranking instances and other methods for selecting instances after ranking them.

Acknowledgments

This work was partly supported by the National Council of Science and Technology of Mexico (CONACyT) under the Projects CB-2008-01-106443, CB-2008-01-106366 and Grant 234507.

References

[1] H. Cheng, J. Shan, W. Ju, Y. Guo, L. Zhang, Automated breast cancer detection and classification using ultrasound images: a survey, Pattern Recognition 43 (2010) 299–317.
[2] S. Rosset, C. Perlich, G. Świrszcz, P. Melville, Y. Liu, Medical data mining: insights from winning two competitions, Data Mining and Knowledge Discovery 20 (2010) 439–468.
[3] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning 46 (2002) 389–422.
[4] S. Sun, C. Zhang, D. Zhang, An experimental evaluation of ensemble methods for EEG signal classification, Pattern Recognition Letters 28 (2007) 2157–2163.
[5] H. Liu, L. Liu, H. Zhang, Ensemble gene selection for cancer classification, Pattern Recognition 43 (2010) 2763–2772.
[6] B. Twala, M. Phorah, Predicting incomplete gene microarray data with the use of supervised learning algorithms, Pattern Recognition Letters 31 (2010) 2061–2069.
[7] N. Kasabov, Global, local and personalised modeling and pattern discovery in bioinformatics: an integrated approach, Pattern Recognition Letters 28 (2007) 673–685.
[8] T. Gestel, B. Baesens, J. Suykens, D. Van den Poel, D. Baestaens, M. Willekens, Bayesian kernel based classification for financial distress detection, European Journal of Operational Research 172 (2006) 979–1003.
[9] H. Ince, T. Trafalis, Kernel methods for short-term portfolio management, Expert Systems with Applications 30 (2006) 535–542.
[10] Y. Zhang, Z. Lu, J. Li, Fabric defect classification using radial basis function network, Pattern Recognition Letters 31 (2010) 2033–2042.
[11] H. Liu, H. Motoda, Instance Selection and Construction for Data Mining, vol. 608, Springer, Netherlands, 2001.
[12] H. Liu, H. Motoda, On issues of instance selection, Data Mining and Knowledge Discovery 6 (2002) 115–130.
[13] J. Olvera-López, J. Carrasco-Ochoa, J. Martínez-Trinidad, J. Kittler, A review of instance selection methods, Artificial Intelligence Review 34 (2010) 133–143.
[14] J. Bezdek, L. Kuncheva, Nearest prototype classifier designs: an experimental study, International Journal of Intelligent Systems 16 (2001) 1445–1473.
[15] A. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artificial Intelligence 97 (1997) 245–271.
[16] T. Reinartz, A unifying view on instance selection, Data Mining and Knowledge Discovery 6 (2002) 191–210.
[17] S. Garcia, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Transactions on Pattern Analysis and Machine Intelligence 99 (2011).
[18] J. Derrac, S. García, F. Herrera, A survey on evolutionary instance selection and generation, International Journal of Applied Metaheuristic Computing (IJAMC) 1 (2010) 60–92.
[19] N. Jankowski, M. Grochowski, Comparison of instances selection algorithms i. Algorithms survey, in: Artificial Intelligence and Soft Computing-ICAISC 2004, 2004, pp. 598–603.
[20] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1967) 21–27.
[21] P.E. Hart, The condensed nearest neighbor rule, IEEE Transactions on Information Theory 14 (1968) 515–516.
[22] C. Chou, B. Kuo, F. Chang, The generalized condensed nearest neighbor rule as a data reduction method, in: 18th International Conference on Pattern Recognition. ICPR 2006, vol. 2, IEEE, 2006, pp. 556–559.
[23] D. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Transactions on Systems, Man and Cybernetics 2 (1972) 408–421.
[24] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Mining and Knowledge Discovery 6 (2002) 153–172.
[25] R. Wilson, T. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning 38 (2000) 257–286.
[26] K. Nikolaidis, J. Goulermas, Q. Wu, A class boundary preserving algorithm for data condensation, Pattern Recognition 44 (2011) 704–715.
[27] I. Czarnowski, Cluster-based instance selection for machine classification, Knowledge and Information Systems (2010) 1–21.
[28] S. García, J.R. Cano, F. Herrera, A memetic algorithm for evolutionary prototype selection: a scaling up approach, Pattern Recognition 41 (2008) 2693–2709.
[29] N. García-Pedrajas, J. Romero del Castillo, D. Ortiz-Boyer, A cooperative coevolutionary algorithm for instance selection for instance-based learning, Machine Learning 78 (3) (2010) 381–420, http://dxdoi.org/10.1007/s10994-009-5161-3.
[30] A. Lumini, L. Nanni, A clustering method for automatic biometric template selection, Pattern Recognition 39 (2006) 495–497.
[31] J. Olvera-López, J. Carrasco-Ochoa, J. Martínez-Trinidad, A new fast prototype selection method based on clustering, Pattern Analysis & Applications 13 (2010) 131–141.
[32] E. Marchiori, Class conditional nearest neighbor for large margin instance selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2010) 364–370.
[33] J. Lin, Divergence measures based on the Shannon entropy, IEEE Transactions on Information Theory 37 (2002) 145–151.
[34] C. Vallejo, J. Troyano, F. Ortega, InstanceRank: bringing order to datasets, Pattern Recognition Letters 31 (2010) 133–142.
[35] L. Page, S. Brin, R. Motwani, T. Winograd, The Pagerank Citation Ranking: Bringing Order to the Web, Technical Report, Stanford Digital Library Technologies Project, 1998.
[36] O. Hamsici, A. Martinez, Sparse kernels for Bayes optimal discriminant analysis, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07, IEEE, pp. 1–7.
[37] M. Tipping, C. Nh, Sparse kernel principal component analysis, in: Advances in Neural Information Processing Systems, vol. 13, 2001.
[38] M. Wu, B. Schölkopf, G. Bakır, A direct method for building sparse kernel learning algorithms, Journal of Machine Learning Research 7 (2006) 603–624.
[39] A. De Haro-García, N. García-Pedrajas, A divide-and-conquer recursive approach for scaling up instance selection algorithms, Data Mining and Knowledge Discovery 18 (2009) 392–418.
[40] C. García-Osorio, A. de Haro-García, N. García-Pedrajas, Democratic instance selection: a linear complexity instance selection algorithm based on classifier ensemble concepts, Artificial Intelligence 174 (5–6) (2010) 410–441.

[41] A. Frank, A. Asuncion, UCI Machine Learning Repository, 2010.
[42] D. Wilson, T. Martinez, Improved heterogeneous distance functions, Journal of Artificial Intelligence Research 6 (1997) 1–34.
[43] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[44] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann Publishers, 2005.
[45] J. Friedman, T. Hastie, R. Tibshirani, The Elements of Statistical Learning, Springer Series in Statistics, vol. 1, 2001.
[46] A. Téllez-Valero, M.M. y Gómez, O. Fuentes-Chavez, L. Villaseñor-Pineda, Automatic classification of texts about natural disasters in Mexico, in: International Congress on Computer Science Research, Oaxtepec, México.

Pablo Hernandez-Leal is a Ph.D. student at the Computer Science Department of the National Institute of Astrophysics, Optics and Electronics (INAOE). He received his B.S. degree from the Autonomous University of Puebla in 2009. In 2011 he obtained his M.Sc. degree from INAOE. His research interests include multiagent systems, probabilistic graphical models and pattern recognition.

José FCO. Martínez-Trinidad received his B.S. and M.Sc. degrees in Computer Science from the Autonomous University of Puebla, Mexico, in 1995 and 1997, respectively, and his Ph.D. degree in the National Polytechnic Institute, Mexico, in 2000. Professor Martinez-Trinidad edited/authored six books and over 100 papers, on subjects related to pattern recognition.

Jesús-Ariel Carrasco-Ochoa received his Ph.D. degree in Computer Science from the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2001. He works as full time researcher at the National Institute for Astrophysics, Optics and Electronics of Mexico. His current research interests include logical combinatorial pattern recognition, data mining, testor theory, feature and prototype selection, text analysis, fast nearest neighbor classifiers and clustering.

José Arturo Olvera López received his B.S. degree in Computer Science from the Faculty of Computer Science of the Autonomous University of Puebla, Mexico, in 2004; his M.Sc. degree in Computer Science from the National Institute of Astrophysics, Optics and Electronics (INAOE), Mexico, in 2006 and his Ph.D. degree in Computer Science from INAOE, Mexico, in 2009. Currently, he is a full time researcher in the Computer Science Department at the Autonomous University of Puebla (BUAP), Mexico. His research interests are logical combinatorial pattern recognition, machine learning and prototype selection.