InstanceRank: Bringing order to datasets

Carlos G. Vallejo *, José A. Troyano, F. Javier Ortega

Department of Computer Languages and Systems, University of Seville, Avda. Reina, Mercedes s/n, 41012 Seville, Spain

Keywords:
Instance-based learning
Instance reduction
Nearest neighbor
PageRank Classification

A B S T R A C T

In this paper we present InstanceRank, a ranking algorithm that reflects the relevance of the instances within a dataset. InstanceRank applies a similar solution to that used by PageRank, the web pages ranking algorithm in the Google search engine. We also present ISR, an instance selection technique that uses InstanceRank. This algorithm chooses the most representative instances from a learning database. Experiments show that ISR algorithm, with InstanceRank as ranking criteria, obtains similar results in accuracy to other instance reduction techniques, noticeably reducing the size of the instance set.

1. Introduction

The nearest neighbor technique obtains good classification results using a very simple algorithm. Strictly speaking, there is no learning stage since no model is built. Each time an instance is classified, it is compared to all other instances contained in the learning database and sorted in categories on the basis of the most similar instances. This allows for the classification algorithm to take into account possible search space regions which would be left out by other algorithms that do perform data-based generalizations. There are not only advantages, however; the cost of having a simple and powerful technique lies in demanding large storage space (given there is no model, all instances need to be memorized) and time (each instance to be classified needs to be compared to all those contained in the dataset) during the classification stage.

In order to solve this problem, there have been proposed several solutions which could be classified into two categories: those directed towards the reduction of the number of instances in the databases without losing classification quality (Aha et al., 1991; Wilson and Martinez, 2000; Brighton and Mellish, 2002; Bezdek and Kuncheva, 2001; Paredes and Vidal, 2006), and those intended to speed up the search for the most-similar exemplars with the help of data and index structures (Papadimitriu and Bentley, 1980; Vidal, 1986; Sproull, 1991; Yianilos, 1993; Brin, 1995; Gómez-Ballester et al., 2006). In our work we show a ranking that characterizes the relevance of points in a database and an instance reduction algorithm using this ranking which falls within the first of the categories above. In spite of appearances, when an instance reduction technique is applied, it not always get worse classification accuracy. In some cases, classification is better in reduced sets than in the original database. In these cases it is said that the technique edits the database since it can reduce noisy instances.

We have named the instance ranking developed in our work "InstanceRank". This ranking is estimated using similar techniques to those used in calculating nodes relevance in a graph. Specifically, we have used an adjustment of the PageRank algorithm which is one of the criterion in which Google websearch engine is based. This ranking has been used in an instance selection algorithm called ISR. This algorithm's results are statistically equivalent to those of the best algorithms of this type regarding accuracy; ISR clearly improves these techniques in instance set size reduction, except one of them (Explore) which has no significant difference with ISR.

The rest of the paper is organized as follows: Section 2 presents InstanceRank; Section 3 contains the ISR algorithm; Section 4 presents the experimental design, results obtained, their statistical analysis and an analysis of the set of instances selected by our algorithm; lastly, Section 5 includes our conclusions and future lines of work.

2. InstanceRank: an instances-ranking criterion

The general idea that has guided our work is the thought that the relevance measure offered by a ranking of instances might be useful to determine which ones are more relevant within a database. A similar idea, TextRank, has been used in the scope of

* Corresponding author. Tel.: +34 954552768; fax: +34 954557139.
E-mail address: vallejo@us.es (C.G. Vallejo).

Natural Language Processing to arrange the different elements of a text according to their relevance.

2.1. PageRank

PageRank algorithm was put forward by Brin and Page (1998), Page et al. (1998). Its foundations can be explained as follows: Internet can be seen as a directed graph in which nodes are made up by the different web pages and links between the pages are the edges. If there is a link between two pages, in the graph there would be an edge going from the node representing the first page to the node representing the second page. Taking into account the topology of this graph, we can calculate a relevance index for each node, this is PageRank. The theoretical substratum of PageRank is the existence of a "random surfer" who visits web pages by clicking the links in them, or by writing a new address in the browser's address bar, going to a new page which is not connected to the ones visited before. The mathematical expression of this is as follows: we name the PageRank of page V as $PR(V)$, the set of pages that have a link to V (which in graph terminology would be the origin nodes to which V is linked) as $In(V)$, and the set of nodes to which V is linked (similarly, the target nodes of the edges which have V as their origin) as $Out(V)$, then

$$PR(V) = (1-d) + d \sum_{W \in In(V)} \frac{PR(W)}{|Out(W)|},$$

where d parameter is a value between 0 and 1 (in (Brin and Page, 1998) the value of 0.85 is recommended) reflecting the probability that the "random surfer" visits pages clicking links, and $1-d$ is the probability that the surfer visits any given page by chance. The idea behind the expression of the PageRank calculation is that there are more probabilities of a page being reached through links, if there are many links from pages that have high probabilities of being visited (having a greater PageRank).

PageRank can be estimated through a simple iterative algorithm and represents a probability distribution over web pages.

2.2. TextRank

TextRank, by Mihalcea and Tarau (2004) is an adaptation of PageRank to a very different context: the natural language processing; more specifically, the extraction of key words and the generation of summaries. The main idea is that a word or phrase (in general, a graph's vertex) has more or less relevance depending on how much it is influenced by the rest; this relevance is depicted by TextRank. The influence of a word or phrase on another depends on their similarity; that is the reason why, as opposed to PageRank, influence has a certain weighting: vertex V_i influences vertex V_j with a certain weight w_{ij} (the graph is therefore weighted).

The expression of PageRank, once the weights are introduced, becomes:

$$TR(V_i) = (1-d) + d \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{v_k \in Out(V_j)} w_{jk}} TR(V_j)$$

The influence of a word or phrase on another is reciprocal and that is why the graph describing this situation is an undirected graph ($w_{ij} = w_{ji}, In(V) = Out(V)$), which simplifies calculations.

The expression of TextRank, as that of PageRank, can be estimated through an iterative calculus that converges.

2.3. InstanceRank

Our proposal consists in considering the instances of a database as the vertexes of a graph, similarly to what PageRank and TextRank do. Each edge of this graph represents the similarity between the instances representing the ends of the edges; this edge is labeled with that similarity. Given its nature, this is a complete and undirected graph. We have defined the similarity between instances as a function of the distance between them; in a sense this is the reverse of the concept of distance. A similarity function $\mathrm{sim}(x,y)$ between the instances of a dataset T should fulfill three properties $\forall v_1, v_2 \in T$:

$$0 \leqslant \mathrm{sim}(v_1, v_2) \leqslant 1$$
$$\mathrm{sim}(v_1, v_1) = 1$$
$$\mathrm{sim}(v_1, v_2) = \mathrm{sim}(v_2, v_1)$$

We have used HOEM (Heterogeneous Euclidean-Overlap Metric) (Wilson and Martinez, 1997) for the distance between two instances. It is the Euclidean distance for the continuous features, adjusted for the maximum and minimum values, and the overlap distance (0 for the ones with the same value and 1 for those with different values) for the discrete features. Lastly, the distance is normalized so that

$$\forall v_1, v_2 \in T \quad 0 \leqslant \mathrm{dist}(v_1, v_2) \leqslant 1$$

We have experimented with the following similarity functions:

$$\mathrm{sim}(v_1, v_2) = 1 - \frac{\mathrm{dist}(v_1, v_2)}{k}$$
$$\mathrm{sim}(v_1, v_2) = \frac{1}{1 + k\,\mathrm{dist}(v_1, v_2)}$$
$$\mathrm{sim}(v_1, v_2) = e^{-k\frac{\mathrm{dist}(v_1, v_2)}{(1-\mathrm{dist}(v_1, v_2))}}$$
$$\mathrm{sim}(v_1, v_2) = e^{-k\,\mathrm{dist}(v_1, v_2)^2}$$

where k ($k > 0$) is a parameter with a value that should be adequately adjusted. These four functions fulfill the three properties above. The graph depicting these functions can be seen in Fig. 1, in which we have represented the values for $k = 1$ and $k = 20$ for each one of them; the rest of the values are included in the area delimited by these two.

The expression of the InstanceRank calculation on the set of instances T, where instances are $\{V_i, 1 \leqslant i \leqslant |T|\}$, is as follows:

$$IR(V_i) = (1-d) + d \sum_{j=1}^{|T|} \frac{\mathrm{sim}(V_j, V_i)}{\sum_{k=1}^{|T|} \mathrm{sim}(V_j, V_k)} IR(V_j). \tag{1}$$

We have carried out experiments with two types of InstanceRank:

GlobalInstanceRank: it does not take into account the class to which each instance belongs to, and it therefore measures the overall density of instances.

LocalInstanceRank: distinctions are made between each of the classes so that the similarity between two instances is zero when both belong to different classes, and its real value if they belong to the same class.

LocalInstanceRank could be interpreted as follows: a higher rank value means that the corresponding instance has greater local density of instances of its same class; a smaller value would mean either that it is a peripheral instance (in the borderline between two classes), or that it is an instance surrounded by instances of another class and that could therefore be a representation of noise.

In general, InstanceRank measures the relevance of an instance within the dataset. It therefore allows us to establish a ranking among instances. We have used *GlobalInstanceRank* in a preliminary work (Vallejo et al., 2007), but in the ISR algorithm *LocalInstanceRank* is used as the criterion to select the instances of the training database because it yields the best results.

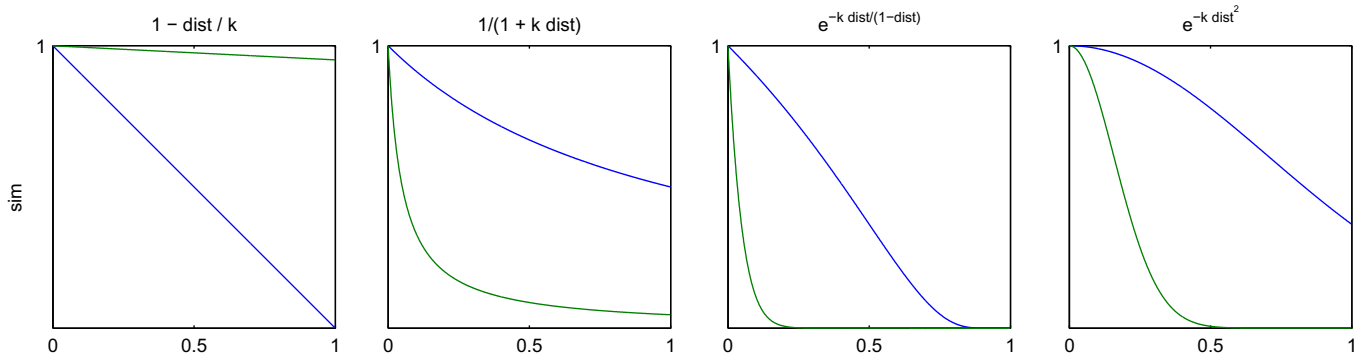

Fig. 1. Similarity functions.

2.3.1. Algebraic justification

We will now see the algebraic justification underlying InstanceRank. At this point, and as a means of simplifying our notation, we will name $IR(V_i)$ as x_i and $N = |T|$. We denote as s_{ij}=sim(V_j, V_i) and $s_j = \sum_{k=1}^{N} s_{kj}$. Then the expression of InstanceRank calculation could be expressed in a matrix form as:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = (1-d) \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + d \begin{bmatrix} s_{11}/s_1 & s_{12}/s_2 & \dots & s_{1N}/s_N \\ s_{21}/s_1 & s_{22}/s_2 & \dots & s_{2N}/s_N \\ \vdots & \vdots & & \vdots \\ s_{N1}/s_1 & s_{N2}/s_2 & \dots & s_{NN}/s_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}.$$

If we name the above square matrix as $\boldsymbol{S}$ and the $N \times 1$ ones vector as $\boldsymbol{u}$, then the expression above could be rewritten as:

$$\boldsymbol{x} = (1-d)\boldsymbol{u} + d\boldsymbol{S}\boldsymbol{x}. \tag{2}$$

Matrix $\boldsymbol{S}$ is a stochastic matrix: on the one hand its values, the quotients $s_{ij}/s_j = s_{ij} \Big/ \sum_{k=1}^{N} s_{kj}$, are obviously within the 0–1 range; on the other hand, the sum of the elements in column i is

$$\sum_j \frac{s_{ji}}{s_i} = \frac{1}{s_i} \sum_j s_{ji} = \frac{1}{s_i} s_i = 1.$$

If $\boldsymbol{x}$ is normalized, so that $\|\boldsymbol{x}\|_1 = N$, that is, the sum of the components is N (if we start from a vector with no negative elements it will always remain non-negative since the elements of $\boldsymbol{S}$ are so), then we have that

$$\boldsymbol{u}^T\boldsymbol{x} = N$$

so that expression (2) of the calculation of $\boldsymbol{x}$ could be written as

$$\boldsymbol{x} = (1-d)\boldsymbol{u}\frac{1}{N}\boldsymbol{u}^T\boldsymbol{x} + d\boldsymbol{S}\boldsymbol{x}$$

or

$$\boldsymbol{x} = ((1-d)\boldsymbol{e}\boldsymbol{u}^T + d\boldsymbol{S})\boldsymbol{x},$$

where $\boldsymbol{e}$ is a vector with elements that are equal to $1/N$.

$\boldsymbol{e}\boldsymbol{u}^T$ is again a stochastic matrix: all of its elements are within the 0–1 range (they are all equal to $1/N$) and its columns add up to 1.

If we name

$$\boldsymbol{M} = ((1-d)\boldsymbol{e}\boldsymbol{u}^T + d\boldsymbol{S})$$

the system is written as

$$\boldsymbol{x} = \boldsymbol{M}\boldsymbol{x}$$

$\boldsymbol{M}$ is the convex combination (linear combination with positive coefficients whose sum is 1) of two stochastic matrices, so $\boldsymbol{M}$ is also stochastic. Its values are within the 0–1 range and it is normalized (its columns add up to 1), therefore, as a result of the Perron–Frobenius theorem, has 1 as an eigenvalue and the rest are lower than 1 in module (they could be complex). The solution to the above system is an eigenvector corresponding to the eigenvalue 1.

2.3.2. Convergence of InstanceRank

Calculating $\boldsymbol{x}$ (that is, InstanceRank) can be done from expression (1) through an iterative process which converges:

If

$$\boldsymbol{x}_{k+1} = (1-d)\boldsymbol{u} + d\boldsymbol{S}\boldsymbol{x}_k$$

and

$$\boldsymbol{x}_k = (1-d)\boldsymbol{u} + d\boldsymbol{S}\boldsymbol{x}_{k-1}$$

then

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| = d\|\boldsymbol{S}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})\| \leqslant d\|\boldsymbol{S}\|\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|$$

for any compatible vector and matrix norm. For example using 1-norm,

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{N} |x_i|,$$

and

$$\|\boldsymbol{S}\|_1 = \max_{j=1,\dots,N} \sum_{i=1}^{N} |s_{ij}|.$$

In this case $\|\boldsymbol{S}\|_1 = 1$, and as $d < 1$, so convergence is guaranteed.

3. ISR: Instance Selection based on Ranking

In this section we present ISR (*Instance Selection based on Ranking*), an instance-based learning algorithm that allows us to assess the usefulness of sorting instances by InstanceRank applied to the issue of selecting instances. The design of our algorithm is rooted in the premise that the ranking offered by InstanceRank may be useful in determining which instances in a database are more relevant, just as PageRank measures the relevance of a web page in the Internet or TextRank of a word or phrase in a text.

Our inspiration in designing ISR derived from the WITS algorithm presented by Morring and Martinez (Morring and Martinez, 2004), an instance selection algorithm depending on the order in which instances are processed; this algorithm considers instances following a criterion known as typicality. This concept was proposed by Zhang (1992) as a measure of an instance's representativity within a class. It is defined as the quotient of the instance's average similarity with the rest of instances in its class (intra-class similarity) and the average similarity with all instances of a different class (extra-class similarity). The similarity between two instances x and y is defined as $1 - dist(x, y)$.

Just as the WITS algorithm is based on typicality, ISR is based on the ranking produced by InstanceRank to determine the order in

which instances are processed. Taking this ranking as its foundation, ISR produces a set S as output in which instances of a database T are weighted. This weight can be interpreted as the radius of the influence sphere of the instance and allows this algorithm to better adjust to the data area (Cost and Salzberg, 1993, 2006). The pseudocode of the ISR algorithm can be seen in Algorithm 1.

Input: T: Set of Instances
Output: S: Reduced set of Instances

```
1  computeInstanceRank(T)                    /* Computes InstanceRank as shown in equation 1 */
2  sort(T)                                   /* Sorts the instances of T in descending InstanceRank order */
3  numClasses := numClasses(T)
4  for i := 1 ... numClasses do bucket_i := EmptyQueue        /* Empty queues of instances */
5  foreach p ∈ T do enqueue(p, bucket_{p.class})  /* Each instance is assigned to the bucket of
   its own class */
6  S := ∅                                    /* The algorithm works incrementally */
7  for i = 1 ... numClasses do bucketNumErrors_i := |bucket_i|   /* S is initially empty, so
   all the instances of the buckets are misclassified by the instances of S */
8  totalNumErrors := |T|
9  repeat
10     j := argmax_i(bucketNumErrors_i)  /* The index of the bucket with the largest number
       of misclassified instances */
11     p := dequeue(bucket_j)
12     pNumErrors := ∞
13     for k = 1 ... 30 do
14         p.weight := b^k                                  /* b has value 1.1 */
15         weightNumErrors := computeErrors(S ∪ p, T)   /* Computes the number of errors
           committed by S ∪ p evaluating the instances of T */
16         if weightNumErrors < pNumErrors then          /* This weight is better */
17             pNumErrors := weightNumErrors
18             bestWeight := p.weight
19         endif
20     endfor
21     if totalNumErrors − pNumErrors ≥ G · |T| then /* This instance, with this weight,
       classifies correctly at least G · |T| more instances of T */
22         p.weigth := bestWeight
23         S := S ∪ p          /* add p to the set of selected instances, with the weight
           calculated */
24         totalNumErrors := pNumErrors                  /* for the next iteration */
25         for i := 1 ... numClasses do bucketNumErrors_i := recomputeErrors(bucket_i, S)
           /* recalculate the number of errors committed using S, with its weights,
           classifying the instances of bucket_i */
26     endif
27     maxBuckect := max_i(|bucket_i|)          /* if it is 0, there are no more instances */
28 until maxBucket = 0 ∨ totalNumErrors = 0
```

Algorithm 1: ISR

The algorithm works incrementally; initially the set of selected instances S is empty and the general idea is to include in it the elements that significantly improve the number of correctly classified instances of the training set T.

The main steps of the algorithm are:

1. The InstanceRank of all the instances of the training set T is calculated (line 1).
2. The training set T is sorted in reversed order of InstanceRank (line 2); we call it relevance order.
3. For each class in the training set T, it is created an empty queue, called *bucket*. Each instance of T is assigned to its corresponding bucket maintaining its relevance order (lines 3–5).
4. The following process is performed iteratively until there are no errors (it is impossible to improve S) or there are no more instances in the buckets:
   (a) The most relevant instance p of the bucket j with the largest number of misclassified instances is selected (lines 10–11).
   (b) A weight for p that minimizes the number of errors when p is added to the selected instances set S, is calculated (lines 13–20).
   (c) If the addition of p (weighted with its best weight), significantly improves S, then p is selected (lines 21–26). The parameter G defines the meaning of a "significant improvement".

In the generalization phase, distances are weighted according to the weights assigned to instances. The nearest neighbor's class ($k = 1$) is chosen as the instance class to be generalized, according to that weighted distance.

Assuming n to be the number of instances of the database, the complexity of the algorithm is as follows: the computation of InstanceRank (line 2) is $O(n^2)$. The loop in line 12 is executed n times and the body of the loop is $O(n)$, so the complexity of the whole algorithm is $O(n^2)$. The storage requirements are also $O(n^2)$.

4. Experimentation

4.1. Databases used and comparison criteria

In order to carry out a consistent research on the goodness of the algorithm we had developed, we have carried out an experimentation with the same 31 databases that was used in the work by Wilson and Martinez (2000); in it they presented, among others, the DROP3 algorithm which has proven to be one of the best algorithms developed up to the moment regarding the family based on the nearest neighbor. Databases were taken from the UCI Repository of Machine Learning Databases, University of California Irvine (Asuncion and Newman, 2007), whose characteristics are well known.

We have compared the results of ISR with the results of the nearest neighbor technique (1NN) and those of all the instance selection techniques reported in (Wilson and Martinez, 2000), we have taken accuracy and reduction values of these algorithms from this paper. In our experiments we have used stratified 10-fold cross validation, which is the most used and standardized measure in the sphere of data mining. We have obtained the accuracy values of the nearest neighbor technique from the IB1 algorithm of Weka implementation (Witten and Frank, 2005). We also implemented the ISR algorithm within the Weka environment using its API, which greatly facilitates writing the classifier and especially its assessment.

We have measured accuracy as the percentage of the number of correct classified instances over the total number of instances in the test set. We have measured size as the quotient (displayed in percentages) between the number of instances selected using the technique and the total number of instances in the corresponding fold of the training set. This means that a smaller numerical value indicates that the set size of selected instances is also smaller, and consequently, that the algorithm has a greater reduction capacity.

4.2. Algorithms description

In this subsection we include a brief description of the instance selection algorithms that ISR is compared to:

IB3 (Aha et al., 1991): an instance x is selected and added to S if the nearest acceptable instance in S has a different class than x. The acceptability is defined by a confidence interval determined by a confident expression. After an instance is accepted it could be rejected if the performance of the classifier becomes significantly poor.
Explore (Cameron-Jones, 1995): it has a growing phase where an instance is retained if it reduces the total cost of the system. There is another phase where an instance is removed if that deletion does not increase the cost. Both phases are repeated iteratively. The cost of the system is the sum of several values that characterizes the impact of misclassified instances, the number of instances kept, etc.
DROP2 (Wilson and Martinez, 2000): the associate set of an instance x is the set of all the instances that have x as one of their k nearest neighbors. The method considers instances from the furthest to nearest enemy (an instance of other class), and then prune the training set according to the rule "Remove x if at least as many of the original training instances would be classified correctly without it".
DROP3 (Wilson and Martinez, 2000): an extension of DROP2 with a preliminary phase in which a rule is applied to eliminate noisy instances.
DROP4 (Wilson and Martinez, 2000): an extension of DROP3 that eliminates "center" points.
DROP5 (Wilson and Martinez, 2000): it goes from closest to furthest enemies, removing noisy instances in order to smooth decision boundaries.
DEL (Wilson and Martinez, 2000): like Explore, it applies the concept of the cost of the system, working in a decremental way as the DROP family do.

4.3. Parameters adjustment

We had to adjust several parameters in the development of algorithm ISR until optimum results were reached: the possible influence of an instance on itself in the calculation of InstanceRank, the value of d in this calculation, the similarity function used and the value of parameter k in these similarity functions. Regarding

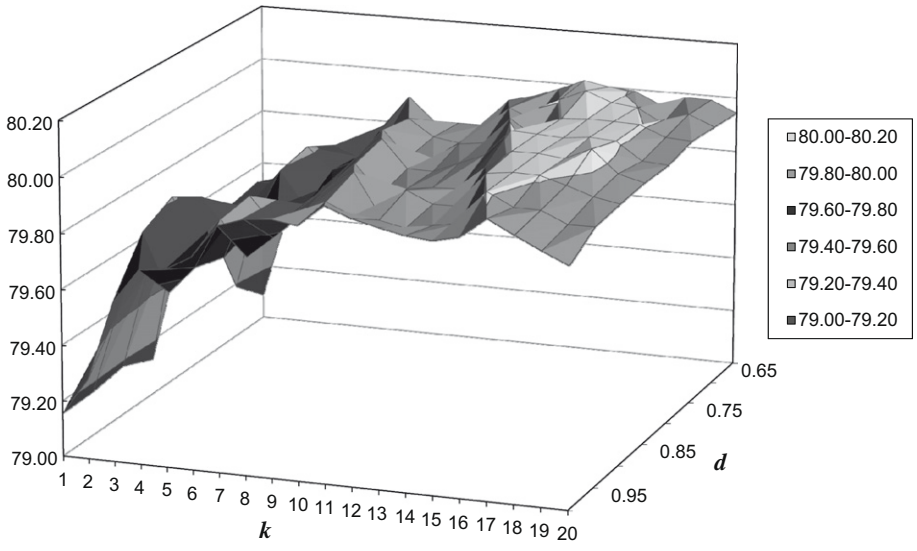

Fig. 2. Accuracy evolution for k and d.

to the influence of an instance on itself, we noted that it was better to use a value close to zero (0 causes convergence problems in classes with only one instance). In order to adjust the other parameters, we carried out experiments for each of the similarity functions with a wide range of k values (1–20) and d values (0.65–1.0, with 0.05 intervals), analyzing in each case the values of these parameters and obtaining the highest accuracy and highest reduction averages in the 31 databases considered. We saw that reduction values remained approximately stable for the best accuracy values, and we therefore took those parameter values yielding the best accuracy average.

Best results were consistently found using the similarity function $\mathrm{sim}(v_1, v_2) = 1 - \mathrm{dist}(v_1, v_2)/k$. In Fig. 2 we can see the evolution of the accuracy depending on the values of parameters k and d for this similarity function, and Fig. 3 shows the corresponding evolution of the set size. We then proceed to analyze the results for this function using smaller intervals and we finally reached the conclusion that the best accuracy values were those for $k = 14.2$ and $d = 0.802$. We have obtained the experimental results using these values for all datasets.

After trying values between 0.05 and 0.00001, we set G, the parameter that decides if an instance improves the classification,

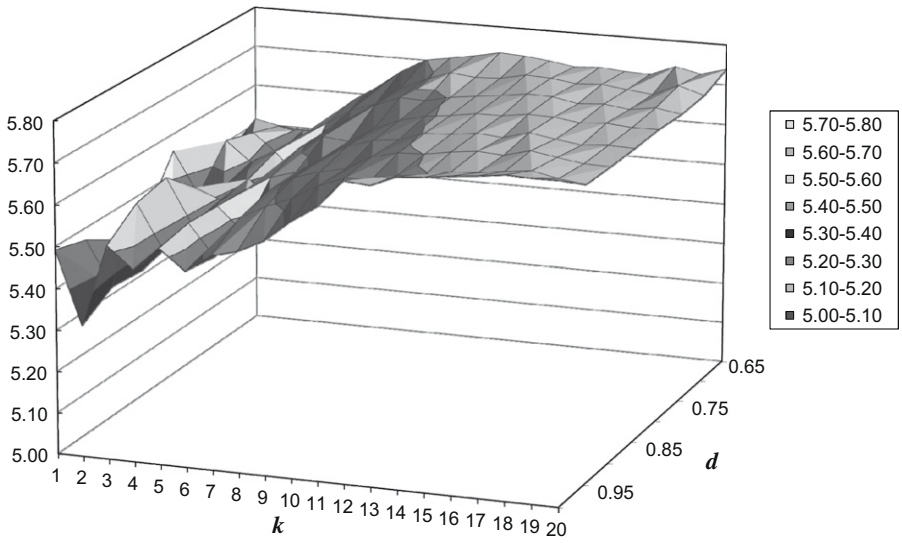

Fig. 3. Set size evolution for k and d.

Table 1
Accuracy of all techniques and ISR.

Database	1NN	IB3	Explore	DROP2	DROP3	DROP4	DROP5	DEL	ISR
Anneal	98.12	91.35	91.11	95.61	94.11	94.36	95.24	93.85	90.98
Australian	82.03	85.22	85.80	83.62	83.91	84.78	83.91	84.78	84.35
Breast cancer (WI)	95.28	96.57	96.71	95.86	96.14	96.28	95.71	96.28	96.14
Bridges	54.72	64.73	57.18	61.18	56.36	57.36	62.82	64.27	64.76
Crx	81.16	86.09	85.51	84.64	85.80	85.51	83.77	83.62	84.78
Echocardiogram	89.19	72.86	94.82	94.82	93.39	94.82	93.39	93.39	90.54
Flags	56.70	49.47	56.16	62.79	61.29	59.58	58.13	56.18	58.25
Glass	70.56	62.14	63.98	65.04	65.02	65.91	65.45	69.59	66.36
Heart	75.19	80.00	81.85	81.85	83.33	81.85	81.11	78.89	83.70
Heart (Cleveland)	76.24	81.16	82.15	79.55	80.84	78.19	79.84	79.49	80.53
Heart (Hungarian)	76.87	79.20	82.30	78.52	80.29	79.22	79.60	77.18	81.97
Heart (Long beach VA)	69.50	70.00	74.50	70.00	73.50	74.00	73.00	70.00	75.00
Heart (More)	72.10	76.31	73.13	73.98	76.38	74.36	74.63	75.15	76.83
Heart (Swiss)	91.06	93.46	93.46	93.46	93.46	93.46	92.63	92.69	93.50
Hepatitis	80.65	73.08	78.67	80.75	81.87	78.75	83.29	80.00	81.94
Horse colic	67.44	66.75	67.09	70.74	70.13	67.73	68.45	67.73	80.16
Image segmentations	93.10	92.14	89.76	92.86	92.62	94.05	89.29	91.90	91.67
Ionosphere	86.32	85.75	80.89	86.60	87.75	86.90	86.90	86.32	91.17
Iris	95.33	94.67	92.67	94.67	95.33	95.33	94.00	93.33	95.33
Led creator + 17	50.84	60.70	72.20	69.20	70.40	69.50	69.80	66.60	42.37
LED creator	61.80	70.40	72.10	71.80	71.70	71.90	72.00	72.30	74.20
Liver (Bupa)	62.90	58.24	57.65	67.77	60.84	62.60	65.50	61.38	62.90
Pima diabetes	70.18	69.78	75.27	70.44	75.01	72.53	73.05	71.61	75.78
Promoters	83.96	91.64	91.36	84.91	86.82	86.82	87.00	83.09	70.75
Sonar	86.54	69.38	70.29	80.88	78.00	82.81	79.88	83.29	83.17
Soybean (Large)	91.21	86.63	85.92	86.60	84.97	86.29	83.73	87.27	81.11
Vehicle	69.86	67.62	60.76	67.37	65.85	67.03	70.22	68.10	63.59
Voting	92.41	95.64	94.25	94.50	95.87	95.87	95.86	94.27	94.25
Vowel	99.43	89.57	57.77	91.08	89.56	90.70	93.36	93.17	71.59
Wine	94.94	91.50	95.46	93.24	94.93	94.93	96.08	94.38	96.07
Zoo	96.67	92.22	95.56	88.89	90.00	91.11	95.56	90.00	96.67
Average	79.75	78.85	79.24	81.07	81.14	81.11	81.39	80.65	80.01

to 0.005 because it obtains the best balance between accuracy and set size. With this value, in small datasets (in datasets under 200 instances) one more instance correctly classified is a good improvement, while in larger ones (for example, Pima) a new instance is accepted if it classifies correctly at least four more instances.

We also adjusted the parameter of the iterative calculation of InstanceRank which indicates the acceptable level of error below which all iterations stop. We took an appropriate value that achieves enough accuracy in the calculations to guarantee the correct sorting of instances. We also analyzed the number of iterations necessary to reach that level of accuracy and results show that they remain at a more than acceptable level (between 3 and 10 iterations).

4.4. Experimental results and statistical analysis

It is hard to balance the size reduction of the set of instances and at the same time to achieve the best accuracy possible. Fig. 4 graphically expresses this trade off. In it we can see the accuracy and the size of the set of instances selected by the algorithms analyzed in (Wilson and Martinez, 2000), and also the values for ISR and 1NN.

The size of the set of instances selected by the algorithm is represented on the horizontal axis, so that values falling more to the left represent algorithms producing smaller, and therefore better, sets of instances selected. For example, the nearest neighbor technique is further to the right since it did not yield a reduction and the set size of instances selected is therefore 100% of the original set of instances.

The accuracy reached by each algorithm is represented on the vertical axis so that the higher an algorithm is depicted the more accurate, and therefore better, it will be. It is therefore desirable to have an instance reduction algorithm depicted on the top left corner of the graph: this indicates that it is a very accurate algorithm with a smaller set size of instances. We can see that algorithms falling in this place are ISR, Explore (Cameron-Jones, 1995), and the algorithms developed by Wilson and Martinez, DROP2, DROP3, DROP4, DROP5 and DEL (Wilson and Martinez, 2000); algorithm IB3 (Aha et al., 1991) is also close by.

We proceed to analyze more closely the results of ISR and these algorithms in the following sections.

4.4.1. Accuracy

Table 1 shows the accuracy values of the techniques that we have compared. ISR obtains the best accuracy results in 11 of the 31 databases analyzed (in three of them it levels scores with another technique).

As compared to the nearest neighbor technique (we have to remember it does not have any reductions), ISR obtains better accuracy results in 19 of the databases; results are leveled in other 3. This shows that ISR has a great power of edition.

Table 2
Ranks of the Friedman test for accuracy values.

Technique	Average rank
1NN	4.06
IB3	4.21
Explore	4.73
DROP2	4.95
DROP3	5.50
DROP4	5.63
DROP5	5.44
DEL	4.53
ISR	5.95

Table 3
Size of the set of instances selected by all techniques and ISR.

Database	1NN	IB3	Explore	DROP2	DROP3	DROP4	DROP5	DEL	ISR
Anneal	100.00	9.79	0.75	8.08	8.65	11.67	9.93	9.30	1.21
Australian	100.00	4.78	0.32	7.28	5.96	7.99	9.18	2.56	0.98
Breast cancer (WI)	100.00	3.47	0.32	3.13	3.58	4.05	4.07	1.89	0.48
Bridges	100.00	28.83	5.67	17.30	17.60	21.28	22.22	35.64	16.40
Crx	100.00	4.28	0.32	7.31	5.46	7.33	7.68	3.08	1.01
Echocardiogram	100.00	11.57	3.01	10.51	10.66	10.96	9.16	6.91	3.60
Flags	100.00	34.14	2.06	20.62	20.45	27.09	25.26	45.88	14.78
Glass	100.00	33.80	3.53	23.10	23.88	29.54	24.81	38.42	20.40
Heart	100.00	13.58	0.82	12.22	13.62	16.71	16.67	4.73	3.62
Heart (Cleveland)	100.00	11.11	0.73	11.92	12.76	15.26	15.37	13.64	3.34
Heart (Hungarian)	100.00	9.90	0.75	8.80	9.86	11.53	11.15	12.28	2.34
Heart (Long beach VA)	100.00	4.89	1.11	11.83	4.50	11.72	14.94	19.28	4.22
Heart (More)	100.00	9.36	0.14	10.71	9.14	13.19	14.62	16.81	0.72
Heart (Swiss)	100.00	3.70	0.90	2.53	1.81	2.35	5.42	4.25	0.90
Hepatitis	100.00	5.09	1.29	10.54	7.81	9.75	9.39	7.59	8.46
Horse colic	100.00	8.49	0.37	8.20	10.30	20.41	14.14	21.82	3.80
Image segmentations	100.00	16.01	2.43	10.45	10.98	12.41	11.35	11.11	3.12
Ionosphere	100.00	14.59	0.63	7.79	7.06	10.60	9.78	12.88	3.99
Iris	100.00	19.78	2.30	14.22	14.81	14.89	12.15	9.56	6.37
LED creator + 17	100.00	32.31	1.40	12.98	12.66	16.37	14.96	20.90	0.29
LED creator	100.00	22.04	1.52	11.85	11.93	13.71	12.33	13.92	1.17
Liver (Bupa)	100.00	10.66	0.64	24.77	24.99	32.56	31.08	38.36	2.96
Pima diabetes	100.00	10.97	0.29	17.59	16.90	21.76	21.95	12.64	1.75
Promoters	100.00	18.12	2.10	13.63	16.67	16.67	12.58	7.34	13.63
Sonar	100.00	12.02	1.07	26.60	26.87	31.20	29.81	29.86	14.21
Soybean (Large)	100.00	30.33	7.78	22.77	25.26	28.41	25.44	24.76	11.15
Vehicle	100.00	28.36	2.47	21.49	23.00	27.88	26.71	32.51	4.36
Voting	100.00	5.44	0.51	4.90	5.11	5.36	7.13	2.02	3.35
Vowel	100.00	36.60	6.65	44.66	45.22	46.02	42.66	36.15	9.26
Wine	100.00	16.60	2.12	11.42	16.11	16.17	9.74	9.05	4.56
Zoo	100.00	29.38	8.40	15.80	20.00	21.60	17.16	18.27	10.74
Average	100.00	16.13	2.01	14.03	14.31	17.30	16.09	16.88	5.72

Table 4
Ranks of the Friedman test and post-hoc analysis for set size values.

	Average rank	$R_i - R_{ISR}$	z	p (uni)	p (bil)
1NN	9.00	6.84	9.83312	< 0.001	< 0.001
IB3	5.77	3.61	5.18970	< 0.001	< 0.001
Explore	1.08	−1.08	−1.55260	0.060	0.121
DROP2	4.31	2.15	3.09082	< 0.001	0.002
DROP3	4.66	2.50	3.59398	< 0.001	< 0.001
DROP4	6.63	4.47	6.42603	< 0.001	< 0.001
DROP5	6.00	3.84	5.52035	< 0.001	< 0.001
DEL	5.39	3.23	4.64342	< 0.001	< 0.001
ISR	2.16	0.00	0.00000		

We can see that there are three remarkable low results: in Led Creator + 17, Promoters and Vowel, accuracy drops more than 16% as compared to DROP3. We have tried to estimate specific sets of parameters for each of them but we do not have achieved a significant improvement respect to the results shown in Table 1.

We have performed a statistical analysis of the results in order to objectively assess the algorithm ISR. We have compared the results of all techniques using the Friedman test, which is the method required to compare the performance of one new technique versus others on different databases (Demšar, 2006, 2008). Friedman test ($\chi_F^2 = 14.871, df = 8, p = 0.062$) shows no significant differences among the accuracies of the techniques analyzed (with $\alpha = 0.05$). Ranks of Friedman test are shown in Table 2. Although there is no significant difference between the techniques, these ranks suggest that the best algorithm, regarding accuracy, is ISR, followed by DROP4, DROP3 and DROP5.

4.4.2. Reduction

Table 3 shows the size of the sets of instances selected by every technique in the datasets. The best reduction ratio is achieved by Explore, followed by ISR.

Friedman test applied to the size values ($\chi_F^2 = 183.744$, $df = 8, p < 0.001$) shows significant differences among the algorithms. The average ranks of Friedman test are shown in Table 4. Note that the rank of ISR is the second after Explore.

Following Demšar (2006) and García and Herrera (2008), we performed a post-hoc analysis (Table 4). The meaning of the columns is as follows: "Avr. Rank" is the Friedman rank, "$R_i - R_{ISR}$" is the difference between the Friedman rank of the technique and the Friedman rank of ISR, "z"$= (R_i - R_{ISR}) \Big/ \sqrt{\frac{k(k+1)}{6N}}$ (where $k = 9$ is the number of techniques and $N = 31$ the number of databases), "p(uni)" is the unilateral p-value for z and "p(bil)" is the bilateral one.

According to the Bonferroni–Dunn correction, differences are significant if p is lower than $\alpha/(k-1)$, that is 0.00625 for $\alpha = 0.05$. ISR is significantly better ($p < 0.001$) than all the other techniques except Explore. However, the difference between ISR and Explore is not significant ($p = 0.121$).

4.4.3. Summary of results and additional remarks

We have shown in the previous sections that Friedman test followed by the post-hoc test establishes that ISR is significantly better in reduction than all the other algorithms analyzed except Explore.

In terms of accuracy, there are no significant differences between ISR and the rest of algorithms, but ranks of Friedman test suggest that the best technique is ISR.

If we compare only Explore and ISR using Wilcoxon signed rank test (Demšar, 2006), we find that ISR is significantly better than Explore in accuracy ($z = 1.861, p = 0.032$) although it is significantly worse in set size ($z = 4.556, p < 0.001$).

It must be taken into account that the results of ISR algorithm (and also 1NN) have been obtained using stratified 10-fold cross validation while for the rest of the algorithms the results were taken from (Wilson and Martinez, 2000), where non stratified 10-fold cross validation was used. There is a slight difference between using stratification or not, but statistical results are exactly the same: ISR non stratified has an average accuracy of 79.32%, and an average set size reduction of 5.51%. Friedman test shows that there are not significant differences among ISR and the others algorithms in accuracy ($\chi_F^2 = 8.839, df = 8, p = 0.356$). Regarding set size, Friedman test shows that there are significant differences ($\chi_F^2 = 183.983, df = 8, p < 0.001$). The post-hoc analysis shows that ISR is better than the other algorithms ($p < 0.001$) except Explore ($p = 0.124$). As the statistical results are the same, we have preferred to maintain the results of ISR using stratification because

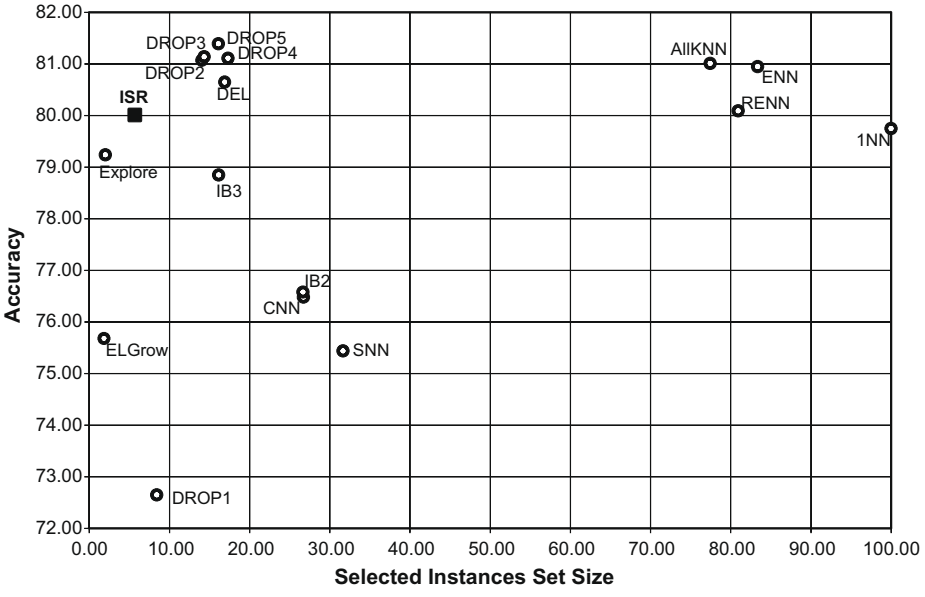

Fig. 4. Accuracy-size balance.

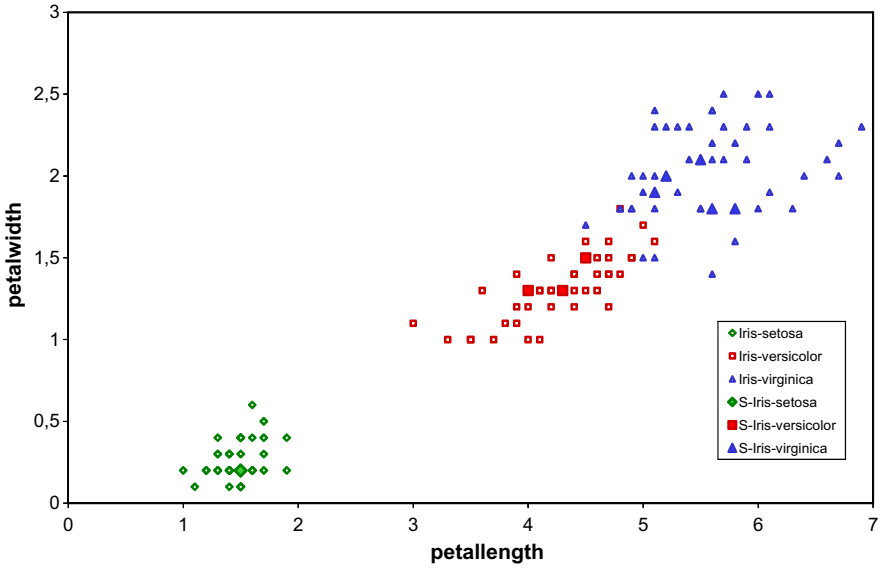

Fig. 5. Instances selected by ISR in Iris.

this can be considered nowadays as the standard in our research field.

We have also performed experiments using a non weighted distance. We have found that the weighted distance gets better accuracy results (80.01% versus 79.39%) although gets slightly worse size reduction results (5.68% versus 5.57%). Statistical analysis obtains the same conclusions using both the weighted and the unweighted distance.

The reduction capacity of the algorithm can be observed, for example, in the well known database Pima. The dataset size obtained with ISR is 1.75% in the average of all 10 folds, which means that there are an average of about 12 elements in the set of instances selected in each fold (as compared to the about 691 elements in each fold of the original training set). The average accuracy achieved with them in the classification of the corresponding test set is 75.78% (nearest neighbor technique obtains 70.18%). Extrapolating this result to the whole set (768 instances), there would be 14 elements in the classifier.

Execution times were in accordance to the complexity of the algorithm ($O(n^2)$), ranging from 360 ms in Zoo (90 instances) to 469 min in Led creator + 17 (10 000 instances). Average time (16.14 min) is strongly biased due to this last database; leaving it out, average time reduces to 60.82 s.

In the parameter adjustment process we chose those values which produced the best results. Anyway, small variations in those parameters do not affect too much the average accuracy or the average size of the reduced sets. For example, for $k = 16.0$ and $d = 1.0$ the average of the accuracies obtained is 79.16%, only 0.85% below the result for the best parameters values, while the average size of the reduced set is 5.65%, even slightly better – 0.03% – than that for the best parameters.

4.5. Sets of data selected

In this subsection we study the nature of instances selected by ISR. We analyze this in the Iris database because it is widely known within the machine learning community.

In Iris, ISR classifies the whole set of data using only an average 6.37% of elements in the original base. If we take into account the whole set of data, then it could be classified using only 9 elements (6% of the elements, or in other words, reaching a reduction of 94% of the original set size). Fig. 5 shows the instances from the Iris database projected over two of its features, *petallength* and *petalwidth*, which are the most representative ones (they were chosen by the feature selection algorithm Correlation-based Feature Subset Selection (CFS) (Hall, 1998)). Blank shapes represent all 150 instances, whereas the black ones are the nine elements selected by ISR, labeled as *S-Iris-setosa*, *S-Iris-versicolor* and *S-Iris-virginica*. ISR has selected one instance from the *Iris-setosa* class, three from the *Iris-versicolor* and five from the *Iris-virginica*. We can see that ISR selects central instances in the clusters formed by different classes; in this dataset they are clearly differentiated. With the instances selected, ISR achieves an accuracy level of 98.67% when it is assessed over its own training set (a resubstitution error of 1.33%).

5. Conclusions and future works

In this paper we have presented a novel instance relevance ranking calculated using an algorithm similar to PageRank. This ranking was used in algorithm ISR, which is an instance selection technique that uses the information offered by the aforementioned ranking to determine the order in which to process the elements of a training database.

Results yielded by the experiments prove that our approach is a very competitive approximation as compared to the most relevant instance reduction algorithms based on the nearest neighbor method. With regard to accuracy ISR achieves the best results in 11 of the databases analyzed, with an average accuracy of 80.01%. The statistical analysis shows that the differences among techniques analyzed are not significant; this is an excellent result taken into account that we have compared ISR to the best instance selection algorithms. In the aspect of set size reduction, our technique obtains excellent results with an average reduction rate of 94.32%. The statistical analysis shows that ISR is significantly better than all the other techniques except Explore, and that the difference between ISR and Explore is not significant. Furthermore, ISR achieves a good balance between both aims: accuracy and size.

We would also like to highlight the editing capacity (or removal of noisy instances) of the technique we have developed: by reducing the number of instances in the classifier, it improves results ob-

tained using the nearest neighbor method in 19 of the 31 databases considered, with equal results in three of the remaining ones.

The most important conclusion is that interpreting the training database through a graph allows us to use graph algorithms (that, as PageRank, have been proven to be effective in other domains) in the field of machine learning.

There are several research directions that look attractive for future exploration. Some of them are: to compare ISR with other families of instance reduction algorithms, to experience with more similarity (and dissimilarity) measures, to apply InstanceRank to other instance selection algorithms, or to analyze the possibilities of using InstanceRank as a means to characterize the internal structure of databases.

Acknowledgements

This research project has been partially financed by the Spanish Ministry of Education and Science (HUM2007-66607-C04-04). The authors wish also to thank two anonymous referees for their helpful suggestions.

References

Aha, D.W., Kibler, D., Albert, M.K., 1991. Instance-based learning algorithms. Mach. Learn. 6, 37–66.

Asuncion, A., Newman, D., 2007. UCI machine learning repository. URL:<http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Bezdek, J.C., Kuncheva, L.I., 2001. Nearest prototype classifier designs: An experimental study. Internat. J. Intell. Syst. 16, 1445–1473.

Brighton, H., Mellish, C., 2002. Advances in instance selection in instance-based learning algorithms. Data Min. Knowl. Disc. 6 (2), 153–172.

Brin, S., 1995. Near neighbor search in large metric spaces. In: Proceedings of the 21st VLDB Conference, pp. 574–584.

Brin, S., Page, L., 1998. The anatomy of a large-scale hypertextual Web search engine. Comput. Networks ISDN Syst. 30 (1–7), 107–117.

Cameron-Jones, R.M., 1995. Instance selection by encoding length heuristic with random mutation hill climbing. In: Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence. pp. 293–301.

Cost, S., Salzberg, S., 1993. A weighted nearest neighbor algorithm for learning with symbolic features. Mach. Learn. 10, 57–78.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

García, S., Herrera, F., 2008. An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. J. Mach. Learn. Res. 9, 2677–2694.

Gómez-Ballester, E., Micó, L., Oncina, J., 2006. Some approaches to improve tree-based nearest neighbour search algorithms. Pattern Recogn. 39 (2), 171–179.

Hall, M.A., 1998. Correlation-based feature subset selection for machine learning. Ph.D. Thesis, University of Waikato, Hamilton, New Zealand.

Mihalcea, R., Tarau, P., 2004. Textrank: Bringing order into texts. In: Lin, D., Wu, D. (Eds.), Proceedings of EMNLP 2004. Association for Computational Linguistics, Barcelona, Spain, pp. 404–411.

Morring, B.D., Martinez, T.R., 2004. Weighted instance typicality search (WITS): A nearest neighbor data reduction algorithm. Intell. Data Anal. 8 (1), 61–78.

Page, L., Brin, S., Motwani, R., Winograd, T., 1998. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

Papadimitriu, C.H., Bentley, J.L., 1980. A worst-case analysis of nearest neighbor searching by projection. Lect. Notes Comput. Sci. 85, 470–482.

Paredes, R., 2006. Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Trans. Pattern Anal. Machine Intell. 28 (7), 1100–1110.

Paredes, R., Vidal, E., 2006. Learning prototypes and distances: A prototype reduction technique based on nearest neighbor error minimization. Pattern Recogn. 39 (2), 180–188.

Sproull, R.F., 1991. Refinements to nearest-neighbor searching in *k*-dimensional trees. Algorithmica 6, 579–589.

Vallejo, C.G., Troyano, J.A., Ortega, F.J., 2007. WIRS: Un algoritmo de reducción de instancias basado en ranking. In: CAEPIA'07: Proceeding of the 12th Conference of the Spanish Association for Artificial Intelligence, pp. 327–336.

Vidal, E., 1986. An algorithm for finding nearest neighbours in (approximately) constant average time. Pattern Recogn. Lett. 4, 145–157.

Wilson, D.R., Martinez, T.R., 1997. Improved heterogeneous distance functions. J. Artif. Intell. Res. 6, 1–34.

Wilson, D.R., Martinez, T.R., 2000. Reduction techniques for instance-based learning algorithms. Mach. Learn. 38, 257–286.

Witten, I.H., Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques, Second edition. Morgan Kaufmann.

Yianilos, P.N., 1993. Data structures and algorithms for nearest neighbor search in general metric spaces. In: SODA '93: Proceedings of the Fourth Annual ACM–SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 311–321.

Zhang, J., 1992. Selecting typical instances in instance-based learning. In: ML92: Proceedings of the Ninth International Workshop on Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 470–479.