PERGAMON

Pattern Recognition 36 (2003) 1009–1018

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Finding representative patterns with ordered projections☆

José C. Riquelme, Jesús S. Aguilar-Ruiz*, Miguel Toro

Department of Computer Science, Technical School of Computer Science & Engineering, University of Seville, Avda. Reina Mercedes s/n, 41012 Seville, Spain

Received 6 February 2001; accepted 10 December 2001

Abstract

This paper presents a new approach to finding representative patterns for dataset editing. The algorithm patterns by ordered projections (POP), has some interesting characteristics: important reduction of the number of instances from the dataset; lower computational cost ($\Theta(mn \log n)$) with respect to other typical algorithms due to the absence of distance calculations; conservation of the decision boundaries, especially from the point of view of the application of axis-parallel classifiers. POP works well in practice with both continuous and discrete attributes. The performance of POP is analysed in two ways: percentage of reduction and classification. POP has been compared to IB2, ENN and SHRINK concerning the percentage of reduction and the computational cost. In addition, we have analysed the accuracy of k-NN and C4.5 after applying the reduction techniques. An extensive empirical study using datasets with continuous and discrete attributes from the UCI repository shows that POP is a valuable preprocessing method for the later application of any axis-parallel learning algorithm.

Keywords: Data mining; Preprocessing techniques; Pattern analysis; Axis-parallel classifiers

1. Introduction

The data mining researchers, especially those dedicated to the study of algorithms that produce knowledge in some of the usual representations (decision lists, decision trees, association rules, etc.), usually make their tests on standard and accessible databases (most of them of small size). The purpose is to verify and validate independently the results of their algorithms. Nevertheless, these algorithms are modified to solve specific problems, for example real databases that contain much more information (number of examples) than standard databases used in training. To accomplish the final tests on these real databases with tens of attributes and thousands of examples is a task that takes a lot of time and memory size.

Among all the methodologies used by data mining researchers, those based on axis-parallel classifiers are the most common. These have an important advantage: they are classifiers that provide easy-to-understand decision rules by humans and are very useful for the expert interested in getting knowledge from the database. The C4.5 tool [1] is probably the most useful technique of this type.

It is advisable to apply to the databases preprocessing techniques to reduce the number of examples or the number of attributes in such a way as to decrease the computational cost. These preprocessing techniques are fundamentally oriented to one of the next goals: editing (reduction of the number of examples by eliminating some of them or finding representative patterns or calculating prototypes) and feature selection (eliminating non-relevant attributes). Our algorithm belongs to the first group.

Editing methods are related to the nearest neighbours (NN) techniques [2]. Some of them are briefly cited in the following lines. Hart [3] proposed to include in the subset S those examples of the training set T whose classification with respect to S are wrong using the NN technique,

☆ This work has been supported by the Spanish Research Agency CICYT under grant TIC2001-1143-C03-02.

* Corresponding author. Depto. de Lenguajes y Sistemas Informaticos, Universidad de Sevilla, Adva. Reina Mercedes s/n, 41012 Seville, Spain.

E-mail address: aguilar@lsi.us.es (J.S. Aguilar-Ruiz).

0031-3203/02/$30.00
PII: S0031-3203(02)00119-X

so that every member of T is closer to a member of S of the same class than to a member of S of a different class; Aha et al. [4] proposed a variant of Hart's method; Wilson [5] proposed to eliminate the examples with incorrect k-NN classification, so that each member of T is removed if it is incorrectly classified with the k nearest neighbours; Tomek [6] extended the idea of Wilson eliminating the examples with incorrect classification from any $i = 1$ to k after completing all k loops; the work of Ritter [7] extended Hart's method and every member of T must be closer to a member of S of the same class than to any member of T. Other variants are based on Voronoi diagrams [8], Gabriel neighbours (two examples are said to be Gabriel neighbours if their diametrical sphere does not contain any other examples) or relative neighbours [9] (two examples p and q are relative neighbours if for all other examples x in the set, is true the expression $dist(p,q) < max\{dist(p,x), dist(q,x)\}$). All of these techniques need to calculate the distances between examples, which is rather time consuming. If n examples with m attributes are considered, the first methods take $\Theta(mn^2)$ time, the Ritter's algorithm is $\Theta(mn^2 + n^3)$; the Voronoi neighbours, Gabriel neighbours and relative neighbours are $\Theta(mn^3)$.

In this paper we present an algorithm, called patterns by ordered projections (POP). The first version of POP, presented in Ref. [10] and called EOP (editing by ordered projection), worked well with continuous attributes. The experiments were carried out using the range-normalised Euclidean metric to compare with the distance-based reduction techniques. Now, we have improved the algorithm to work simultaneously with continuous and discrete attributes, conserving the properties of the initial approach. Therefore, the experiments described in this paper were run using the HOEM distance (see Section 3).

The algorithm has several important characteristics:

- Considerable reduction of the number of examples.
- Lower computational cost $\Theta(mn \log n)$ than other algorithms.
- Absence of distance calculations.
- Conservation of the decision boundaries, especially interesting for applying classifiers based on axis-parallel decision rules (like C4.5).

We have dealt with several databases from the UCI repository [11] (University of California at Irvine). To show the performance of our method we have used k-NN and C4.5 before and after applying POP. Among the most known editing methods, we have chosen IB2 [4], ENN [12] and SRHINK [13]. A 10-fold cross-validation for each method is achieved to reduce the databases. Afterwards, we have used the 1-NN to show the classification accuracy of the reduced sets using the original tests. In addition, C4.5 generates the decision trees from the reduced sets and they are proved with the original tests. Several tables with computational costs, percentages of reduction and classification accuracy for 1-NN, 3-NN, 5-NN and C4.5 are summarised in the experiment section.

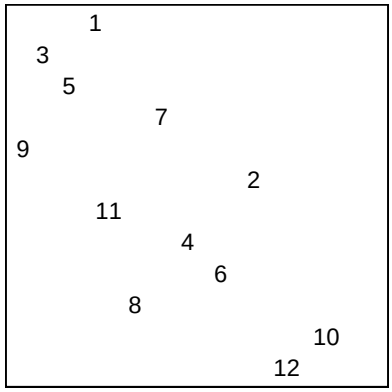

Fig. 1. An example of database.

2. Algorithm

If we choose a region [1] where all examples inside have the same class, perhaps we could select some of them, which are not decisive, in order to establish the boundaries of the region. For example, in two dimensions we need four examples to determine the boundaries of one region, maximum. In general, in d-dimensions we will need $2d$ examples, maximum. Therefore, if a region has more than $2d$ examples, we could reduce the number of them.

That is the main idea of our algorithm: to eliminate the examples that are not in the boundaries of the regions to which they belong. The aim is to calculate which set of examples could be covered by a "pure" region and then eliminate those inside that are not establishing the boundaries. A region is pure if all the examples inside have the same class.

The method is completely heuristic because POP will independently work with the projection of the example in each dimension, not all dimensions at the same time. This heuristic could seem poor for lack of generality; however, the results are quite the opposite.

To show graphically (Fig. 1) the idea of our algorithm we use a simple two-dimensional database with 12 numbered examples and two labels: I (odd numbers) and P (even numbers). An optimal classifier would obtain the two rules shown in Fig. 2. However, this classifier must be hierarchical, since it is producing overlapped rules. This is not the case of C4.5 and many others. An axis-parallel classifier might provide one of the following solutions presented in Figs 3–6, where rules are not overlapped.

Before formally exposing the algorithm, we will briefly explain the main idea. Consider the situation depicted in Fig. 7: the projection of the examples on the abscissa axis

[1] In this context, region means hyper-rectangle.

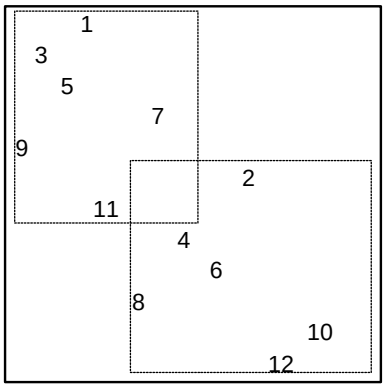

Fig. 2. The best solution with overlapped rules.

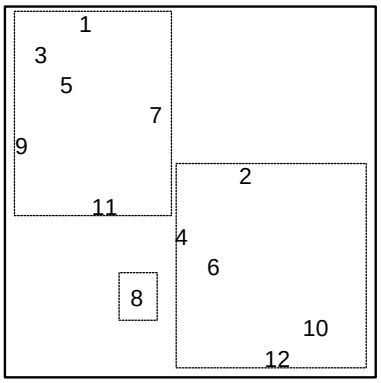

Fig. 3. One possible solution.

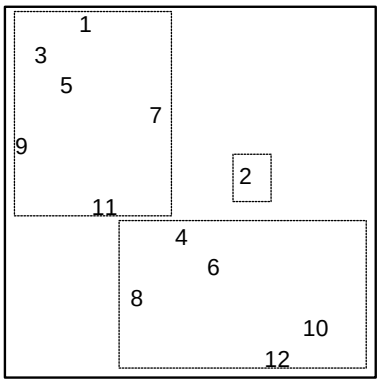

Fig. 4. One possible solution.

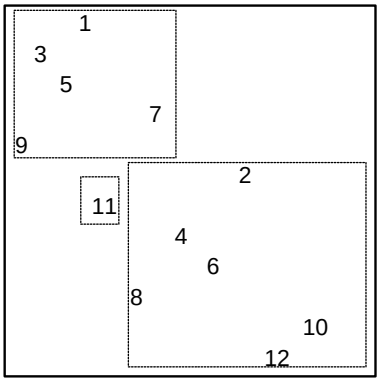

Fig. 5. One possible solution.

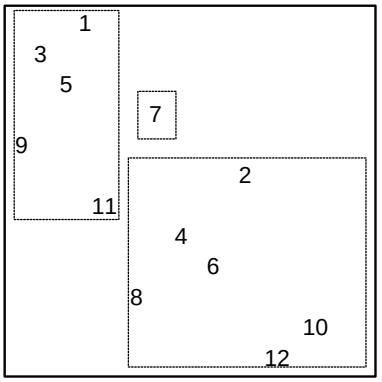

Fig. 6. One possible solution.

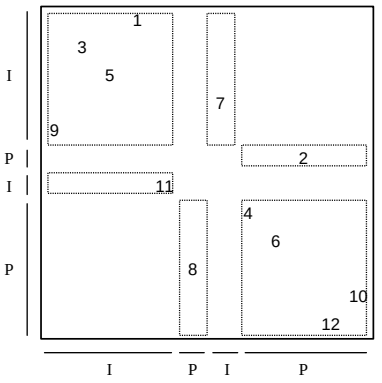

Fig. 7. Regions without overlapping.

produces four ordered sequences $\{I,P,I,P\}$ corresponding to the examples $\{[9,3,5,1,11],[8],[7],[4,6,2,12,10]\}$. Identically, with the projection on the ordinate axis, we can obtain the sequences $\{P,I,P,I\}$ formed by the examples $\{[12,10,8,6,4],[11],[2],[9,7,5,3,1]\}$. Each sequence represents a rectangular region as a possible solution of a classifier (a rule) and the initial and final examples of the sequence (if it has only one, it is simultaneously the initial and the final one) represent the lower and upper values for each coordinate of this rectangle. For example, in Fig. 5, there is a rectangle formed by the examples $\{1,3,5,7,9\}$. This region needs the examples $\{9,7\}$ to establish the

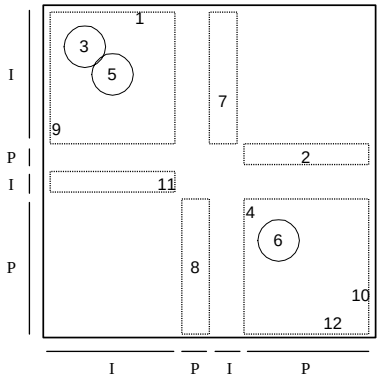

Fig. 8. Result of applying POP.

boundaries of a dimension and the examples $\{1, 9\}$ for another one. Therefore, the remaining examples will be candidates to be eliminated because they are never boundaries. The idea is best understood by analysing the non-empty regions obtained by means of projections on every axis, as shown in Fig. 7 and deleting the examples that are not relevant so as to establish the boundaries of a rule (Fig. 8).

2.1. Definitions

Definition 1. Let the attribute A_i be a real variable that takes values in $I_i = [\min_i, \max_i]$. Then, A is the attributes space defined as $A = I_1 \times I_2 \times \cdots \times I_m$, where m is the number of attributes.

Definition 2. An example $e \in E$ is a tuple formed by the Cartesian product of the value sets of each attribute and the set C of labels. We define the operations *att* and *lab* to access the attributes and its label (or class):

$att : E \times N \to A$ and $lab : E \to C$,

where N is the set of natural numbers.

Definition 3. Let the universe U be a sequence of examples from E. We will say that a database with n examples, each of them with m attributes and a class, forms a particular universe. Then $U = \langle u[1], \ldots, u[n] \rangle$ and as the database is a sequence, the access to an example is achieved by means of its position. Likewise, the access to jth attribute of the ith example is made by $att(u[i], j)$, and for knowing its label $lab(u[i])$.

Definition 4. An ordered projected sequence is a sequence formed by the projection of the universe onto the ith attribute. This sequence is sorted out in increasing order and it contains the numbers of the examples. For example, in Fig. 1, for the first attribute we have $\{9, 3, 5, 1, 11, 8, 7, 4, 6, 2, 12, 10\}$ and for the second attribute $\{12, 10, 8, 6, 4, 11, 2, 9, 7, 5, 3, 1\}$.

Definition 5. A partition in subsequences is the set of subsequences formed from the ordered projected sequence of an attribute in such a way as to maintain the projection order. All the examples belonging to a subsequence have the same class and every two consecutive subsequences are disjointed with respect to the class. In Fig. 7, we have for the first attribute $\{[9, 3, 5, 1, 11], [8], [7], [4, 6, 2, 12, 10]\}$ and for the second attribute $\{[12, 10, 8, 6, 4], [11], [2], [9, 7, 5, 3, 1]\}$. Henceforth, a subsequence will be called a partition.

Definition 6. If an example is in the left or right extreme of a partition, the example is called border. If the partition only has one example, it is a border. The remainders are not border, but inner. For example, in the partition obtained in the previous definition, the examples 9, 11, 8, 4 and 10 are borders for the first attribute.

Definition 7. The weakness of an example is defined as the number of times that example is not a border in a partition (i.e., it is inner to a partition) for every partition obtained from ordered projected sequences of each attribute. In the previous example, let $\{[9, 3, 5, 1, 11], [8], [7], [4, 6, 2, 12, 10]\}$ and $\{[12, 10, 8, 6, 4], [11], [2], [9, 7, 5, 3, 1]\}$ be the partitions, the weakness of each example is given by

$$\begin{aligned} &\text{weakness} = 0 \to \text{examples}\{9, 11, 4\},\\ &\text{weakness} = 1 \to \text{examples}\{1, 8, 7, 2, 12, 10\},\\ &\text{weakness} = 2 \to \text{examples}\{3, 5, 6\}. \end{aligned}$$

Definition 8. Those examples whose weaknesses are equal to the number of attributes of the database are called irrelevant. In our example, there are three irrelevant examples: $\{3, 5, 6\}$, and they do not appear in the solution (Fig. 8).

2.2. Algorithm

The algorithm is conceptually very simple (see Fig. 11). It has two different parts to handle: both continuous and discrete attributes. The number of continuous attributes is m_1 and of discrete attributes is m_2, so that the total number of attributes m is the sum of m_1 and m_2.

The computational cost of POP is $\Theta(m_1 n \log n + m_2 n r)$, where r is the average of the number of different discrete values. Since r might be less than $\log n$ and $m \geqslant max\{m_1, m_2\}$, the cost could be rewritten as $\Theta(mn \log n)$. This cost is much lower than that of other algorithms proposed in the bibliography, normally $\Theta(mn^2)$.

2.2.1. Continuous attributes

The continuous attributes need special treatment due to the sorting. To sort the database in increasing order by an attribute is a task achieved by the QuickSort [14] algorithm. This algorithm is $\Theta(n \log n)$, on average.

After applying QuickSort, we might have repeated values with different class. For this reason, the algorithm firstly sorts by value and, in case of equality, by class. In spite of

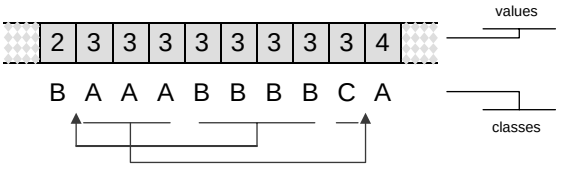

Fig. 9. Quicksort.

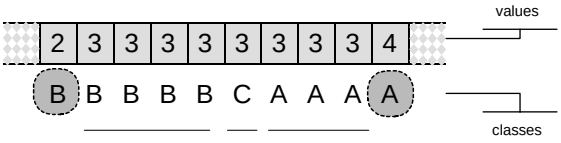

Fig. 10. Resort.

two comparisons, we could find the situation depicted as in Fig. 9.

Despite sorting, the examples sharing the same value for an attribute are not nearer to those examples that have the same class and have another value. In Fig. 9 we can observe that it might be more interesting to have the examples with value 3 and class B nearer to the example with value 2 and class B. The solution to that problem consists of resorting the interval containing repeated values. The heuristic is applied to obtain the least number of changes of class. In this way, the resorting method would produce the output shown in Fig. 10 from the example in Fig. 9.

We have considered the two different values (2 and 4) as pivots for resorting. Then, every example with the same class as the left pivot is moved to the left, and every example with the same class as the right pivot is moved to the right, looking for the adjacent partition. In the middle, the examples will remain with the order generated by QuickSort. That is the algorithmic principle of the method implemented in the ReSort algorithm. The complexity of the ReSort algorithm is $\Theta(n)$, due to the shifting of equal-valued examples. Therefore, the average computational cost of this part of the algorithm is $\Theta(m_1 n \log n)$ (Fig. 11).

2.2.2. Discrete attributes

Due to the nature of discrete attributes, these do not need to be sorted. The set of patterns must be representative and should therefore contain at least an example with every different discrete value. The aim is, increasing the weakness of all the examples except one which has the least weakness obtained for the continuous attributes. The average complexity of this part is $\Theta(m_2 nr)$, where r is the average of the number of different discrete values.

3. Results

Tests have been achieved over 19 databases of varying complexity from the UCI repository [11]. A summary of the characteristics of these databases appears in the appendix.

```
Procedure POP(in: E, out: E)
  for each example e_i ∈ E, i ∈ {1,...,n}
    weakness(e_i):=0
  end for
  for each continuous attribute a_j, j ∈ {1,...,m_1}
    E_j:=QuickSort(E_j, a_j) in incr. order
    E_j:=ReSort(E_j)
    for each example e_i ∈ E_j, i ∈ {1,...,n}
      if e_i is not border
        weakness(e_i):=weakness(e_i)+1
      end if
    end for
  end for
  for each discrete attribute a_j, j ∈ {1,...,m_2}
    for each value v_i^j ∈ a_j
      V:={e | value(e, a_j)=v_i^j }
      Let ē be an example such that weakness(ē) = min_{e∈V}{weakness(e)}
      for each e_i ∈ V except ē
        weakness(e_i):=weakness(e_i)+1
      end for
    end for
  end for
  for each example e_i ∈ E, i ∈ {1,...,n}
    if weakness(e_i)=m
      remove e_i from E
    end if
  end for
end POP
```

Fig. 11. POP algorithm.

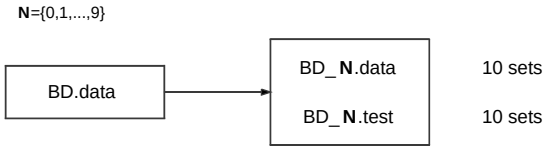

Fig. 12. 10-fold cross-validation.

In our experiments, they all use the heterogeneous overlap-Euclidean metric (HOEM) [5]. This function defines the distance between two values x and y of a given attribute a as in Eq. (1).

$$d_a(x,y) = \begin{cases} rn\_diff(x,y) & \text{if } a \text{ continuous,} \\ overlap(x,y) & \text{if } a \text{ nominal.} \end{cases} \quad (1)$$

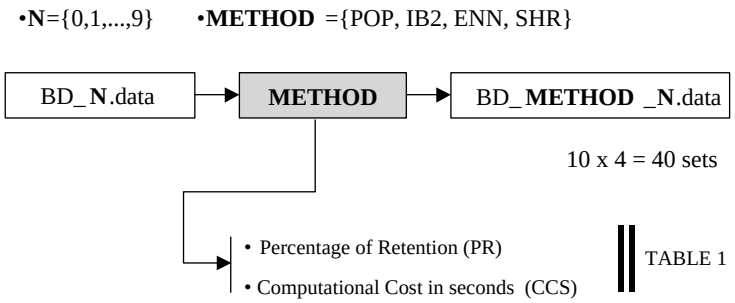

Fig. 13. Reduction methods.

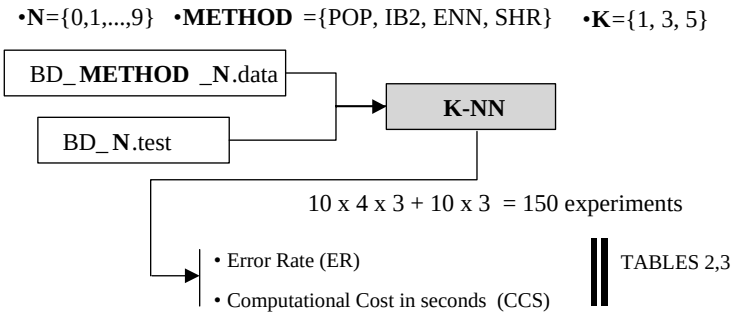

Fig. 14. Classifying with $\{1,3,5\}$-NN.

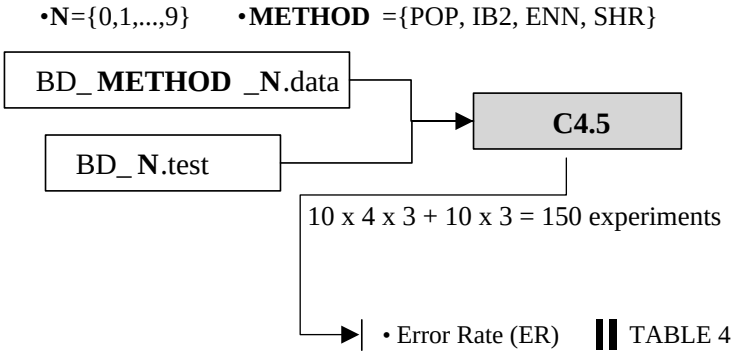

Fig. 15. Classifying with C4.5.

Table 1
Computational cost in seconds (CCS) and percentage of retention (PR) of the editing methods: POP, IB2, ENN and SHRINK. Average is the mean of all databases

Database	POP		IB2		ENN		SHRINK	
	CCS	PR	CCS	PR	CCS	PR	CCS	PR
Bupa	0.7	79.3	1.7	44.3	7.3	62.5	5.5	45.3
Cancer	2.8	21.2	1.9	8.3	38.9	95.8	22.4	6.1
Cars	1.1	76.8	0.7	7.4	10.9	99.4	6.3	4.3
Cleve	1.8	91.0	1.4	30.6	8.9	75.4	6.1	32.3
German	13.4	70.5	25.2	37.5	129.5	72.1	90.7	37.0
Hayes-roth	0.19	91.33	0.22	47.65	1.1	68.26	0.82	53.12
Heart	1.7	92.5	1.4	32.8	7.9	75.5	5.5	32.4
Hepatitis	1.4	83.7	0.5	31.0	3.0	80.3	2.1	27.5
Ionosphe	6.3	99.1	3.8	20.2	32.4	86.7	19.8	16.3
Iris	0.2	66.1	0.1	12.8	1.0	95.3	0.7	10.1
Led7	23.1	9.3	87.0	37.8	467.0	62.7	354.2	27.6
Letter	331.7	34.8	4720.3	13.8	56629.8	96.1	33371.7	8.8
Lenses	0.0	73.6	0.0	51.0	0.0	69.0	0.0	38.9
Monk1	1.1	11.0	1.4	29.6	8.1	73.3	5.9	29.2
Mushroom	396.6	2.9	30.4	0.4	8426.1	100.0	4773.1	0.2
Pima	2.4	74.9	8.1	38.1	45.0	71.2	32.4	37.2
Tic-tac-toe	4.9	8.0	6.8	22.0	53.3	80.4	35.8	21.3
Vote	4.3	25.0	1.3	12.7	19.4	92.5	12.0	9.3
Zoo	0.7	89.4	0.1	12.3	1.2	96.8	0.6	6.9
Average	41.8	57.9	257.5	25.8	3467.9	81.8	2039.2	23.4
Average*	3.9	62.5	8.3	28.0	49.1	79.8	35.3	25.6

* Is the mean without mushroom and letter databases.

Table 2
Computational cost in seconds (CCS) and error rate (ER) of NN, POP, IB2, ENN and SHRINK using 1-NN technique

Database	k-NN		POP		IB2		ENN		SHRINK	
	CCS	ER	CCS	ER	CCS	ER	CCS	ER	CCS	ER
Bupa	0.8	36.6	0.7	38.0	0.4	40.3	0.5	37.4	0.4	38.6
Cancer	4.4	4.0	0.9	6.6	0.4	6.7	4.1	3.9	0.3	21.5
Cars	1.3	0.5	1.0	0.5	0.1	1.3	1.2	1.0	0.1	16.6
Cleve	1.2	23.8	0.9	24.5	0.3	30.4	0.7	18.5	0.3	32.7
German	14.4	28.4	10.0	30.4	5.4	33.8	10.2	27.9	5.3	39.3
Hayes-roth	0.09	32.0	0.09	32.0	0.05	39.5	0.09	48.4	0.06	36.5
Heart	0.9	23.3	0.8	23.7	0.3	26.7	0.7	20.7	0.3	28.5
Hepatitis	0.3	20.0	0.3	20.0	0.1	29.6	0.3	15.4	0.1	36.0
Ionosphe	3.6	12.6	3.5	12.9	0.7	16.9	3.1	14.6	0.6	18.9
Iris	0.1	4.7	0.1	6.0	0.0	8.0	0.1	4.7	0.0	14.0
Led7	50.0	45.5	4.8	60.4	18.9	45.5	31.2	38.6	13.9	84.3
Lenses	0.0	30.0	0.0	61.7	0.0	51.7	0.0	51.7	0.0	40.0
Letter	6471.1	4.0	2168.2	7.0	830.8	7.9	6045.4	4.9	549.7	27.6
Monk1	0.9	25.9	0.1	34.0	0.3	25.2	0.6	29.8	0.3	25.0
Mushroom	949.4	0.0	25.4	0.2	3.8	0.0	957.4	0.0	1.9	28.7
Pima	4.9	28.3	3.7	30.5	2.1	35.8	3.5	24.4	1.9	39.7
Tic-tac-toe	5.8	20.3	0.5	30.2	1.3	19.9	4.6	21.9	1.3	19.4
Vote	2.0	8.0	0.5	7.6	0.3	9.9	2.0	8.0	0.2	52.5
Zoo	0.1	3.0	0.1	6.0	0.0	7.0	0.1	5.0	0.0	54.1
Average	395.3	18.5	116.9	22.7	45.5	22.9	371.9	19.8	30.3	34.4

Table 3
CCS and ER of NN, POP, IB2, ENN and SHRINK using $\{3,5\}$-NN technique

Database	k-NN		POP		IB2		ENN		SHRINK	
	CCS	ER	CCS	ER	CCS	ER	CCS	ER	CCS	ER
3-NN	392.1	17.2	116.8	20.2	46.1	25.6	378.2	18.3	31.5	40.4
5-NN	393.8	16.8	118.3	21.6	46.6	26.0	378.4	18.9	30.9	42.0

Table 4
C4.5: Error rate of the original training files, and the reduced datasets from POP, IB2, ENN and SHRINK

Database	Original	POP	IB2	ENN	SHRINK
Bupa	33.4	37.7	43.8	33.7	41.2
Cancer	6.7	6.3	13.0	5.7	48.5
Cars	1.5	1.3	4.8	2.0	31.4
Cleve	23.5	22.5	29.1	20.5	31.7
German	32.9	34.3	34.2	31.4	39.7
Hayes-roth	18.3	18.3	16.7	23.5	12.0
Heart	25.5	22.9	32.6	20.7	35.2
Hepatitis	20.8	22.0	22.8	21.4	33.1
Ionosphe	11.2	11.4	25.4	9.1	48.0
Iris	4.0	6.0	26.0	6.0	39.3
Led7	27.0	31.6	27.8	37.7	81.0
Lenses	30.0	18.3	31.7	23.3	38.3
Letter	12.1	18.0	27.7	12.5	50.4
Monk1	3.7	29.6	9.2	19.0	24.5
Mushroom	0.0	0.2	18.8	0.0	59.3
Pima	26.6	28.8	31.9	25.0	36.3
Tic-tac-toe	14.2	35.1	23.9	16.1	24.9
Vote	6.2	5.7	7.8	5.0	67.1
Zoo	7.0	8.0	35.8	10.0	82.0
Average	16.0	18.8	24.4	17.0	43.4

The function overlap and the range-normalised difference rn_diff are defined in Eqs. (2) and (3), respectively.

$$\text{overlap}(x,y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{otherwise,} \end{cases} \tag{2}$$

$$rn\_diff(x,y) = \frac{|x-y|}{\max - \min}, \tag{3}$$

where max and min are the upper and lower bounds of the range of the attribute, respectively. And the overall distance between the two input vectors x and y is given by the Eq. (4).

$$HOEM(x,y) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)}. \tag{4}$$

For each database (BD), 10-fold cross-validation was used. A ten-fold cross-validation is performed by dividing the data into 10 blocks of cases that have approximately similar size and for each block in turn, testing the model constructed from the remaining nine blocks on the unseen cases in the hold-out block (Fig. 12).

Each reducing method was given a training set (BD_N) consisting of 90% of the available data, from which it returned a subset BD_METHOD_N (Fig. 13).

The remainder 10% of the unseen data (BD_N.test) was tested on the instances in BD_METHOD_N.data using k-NN, with $k = 1, 3, 5$. The classification over the original (non-reduced) BD_N.data is also reported (see 10×3 in Fig. 14).

As a further comparison, another widely-used learner, C4.5, was run on these datasets (Fig. 15). The purpose is to demonstrate that POP is more useful than other methods if we are interested in producing axis-parallel-based models like that of C4.5 (decision trees), COGITO [15] (decision lists), and many others. POP conserves the axis-parallel decision boundaries, and IB2, ENN and SHRINK do slightly worse.

The experiments show that applying POP, the knowledge in the original training file is conserved into the reduced training file since the decision boundaries of every region in the space are conserved.

A summary of the results of the editing methods appears in Table 1. The first column (CCS) shows the computational cost in seconds of the complete 10-fold cross-validation (the sum of the 10 experiments). The second column (PR) presents the average percentages of examples in the original training file that were included in the reduced file (see Fig. 13). As catalogued in the last row, the most salient aspect is the large difference in time between POP and the other methods when the database has a large number of examples (for example, letter database).

From the point of view of the computational cost, POP outperforms the remainder techniques (Table 1). As the overall averages at the foot of the table indicate, with respect to the computational cost, POP is six times faster than IB2. Without taking into account the mushroom and letter databases (two possible outliers), the time consumption is half as slow for IB2 as it is for POP (see average* in Table 1). ENN and SHRINK are much slower, about ten times. SHRINK and IB2 produces the best percentages of reduction at the expense of increasing the error rate.

In Tables 2 and 3, the results of the classification using k-NN are shown (see Fig. 14). The results obtained by 1- , 3-

Table 5
Global comparison

Method	Time	Retention	Error 1-NN	Error c4.5
POP	41.8	57.9	22.7	18.8
IB2	257.5	25.8	22.9	24.4
ENN	3467.9	81.8	19.8	17
SHRINK	2039.2	23.4	34.4	43.4

and 5-NN are very similar, which shows that the reduction methods conserve the knowledge from the database. ENN, IB2 and POP reach an interesting error rate, similar to NN without reduction, even tough the percentage of reduction of ENN is very low (the resulting database is almost the same, without outliers). SHRINK has a marked increase in error (34.4%). Consequently, it is not a good method for a further classification. As IB2 produces smaller reduced sets, the cost of classifying using k-NN is logically lower.

The results presented in Table 4 (obtained as described in Fig. 15) indicate that IB2 has a weaker performance when C4.5 is used. Using C4.5, POP is more accurate than IB2. When the database is reduced with IB2, it does not conserve the necessary examples so that a parallel classifier, like C4.5, can generate accurate decision trees. POP has an error rate of 18.8%, IB2 has 24.4%. ENN is more accurate than POP, but it takes a lot of time in comparison, as we observed in Table 1.

For the purposes of global comparison, in Table 5 we present the average results of the methods used in this paper. POP has lower cost than the other methods. IB2 and SHRINK reach a great percentage of reduction, although SHRINK has a very high error rate as with NN as C4.5. Applying C4.5 to the reduced database generated by IB2, the decision tree produces an error rate greater than that of POP does.

The comparison clearly indicates that POP is a robust method to reduce databases since it takes a reasonable time and produces accurate results for both k-NN and C4.5. In addition, it is completely deterministic and it does not depend on the order of presentation of examples, like IB2, ENN, SHRINK and many others. This means that it may be more convenient to use POP as a preprocessing method when we are interested in proving the accuracy of a learning method based on axis-parallel representation.

4. Conclusions

In this paper an editing algorithm (POP: patterns by ordered projection) is presented. Its main application is as a preprocessing method for axis-parallel classifiers (like C4.5). POP has an important characteristic: it does not need distance calculations and, therefore, it is not necessary to define it. NN-based techniques need to initially set some parameters, POP does not. The computational cost is lower than other methods $\Theta(mn \log n)$. The test set was carried out with nineteen different databases with continuous and discrete attributes from the UCI repository and the results are very interesting because they show that our algorithm is a robust method to reduce datasets for studying other learning algorithms without loosing decision boundaries. POP is deterministic; it is neither dependent on random values nor the order of example processing.

At the same time, we are presenting a measure, named weakness of an example, which can help to determine the importance of an example as decision boundary. More weakness implies less relevance. Thus, in more complicated databases we could relax the reduction factor for eliminating those examples whose weakness is less than m.

Table 6
Datasets

Database	# Examples	# Nominal	# Continuous	# Classes
Breast cancer	699	0	9	2
Bupa liver disorder	345	0	6	2
Cars	392	1	7	3
Cleveland	303	7	6	2
German	1000	13	7	2
Hayes-Roth	132	0	5	3
Heart disease	270	0	13	2
Hepatitis	155	13	6	2
Ionosphere	351	0	34	2
Iris	150	0	4	3
Led7	3200	7	0	10
Lenses	24	4	0	3
Letter	20,000	0	16	26
Monk1	432	6	0	2
Mushroom	8124	22	0	2
Pima Indian diabetes	768	0	8	2
Tic-Tac-Toe	958	9	0	2
Vote	435	16	0	2
Zoo	101	16	0	7

Appendix

Table 6 lists the number of examples, number of nominal attributes, and number of continuous attributes in each dataset, along with the number of output classes.

References

[1] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
[2] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory IT-13 (1) (1967) 21–27.

[3] P. Hart, The condensed nearest neighbor rule, IEEE Trans. Inf. Theory 14 (3) (1968) 515–516.
[4] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learning 6 (1991) 37–66.
[5] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, J. Artif. Intell. Res. 6 (1) (1997) 1–34.
[6] I. Tomek, An experiment with the edited nearest-neighbor rule, IEEE Trans. Syst. Man Cybern. 6 (6) (1976) 448–452.
[7] G. Ritter, H. Woodruff, S. Lowry, T. Isenhour, An algorithm for a selective nearest neighbor decision rule, IEEE Trans. Inf. Theory 21 (6) (1975) 665–669.
[8] V. Klee, On the complexity of d-dimensional voronoi diagrams, Arch. Math. 34 (1980) 75–80.
[9] G.T. Toussaint, The relative neighborhood graph of a finite planar set, Pattern Recognition 12 (4) (1980) 261–268.
[10] J.S. Aguilar, J.C. Riquelme, M. Toro, Data set editing by ordered projection, in: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'00), Berlin, Germany, 2000, pp. 251–255.
[11] C. Blake, E.K. Merz, UCI repository of machine learning databases, 1998.
[12] D. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421.
[13] D. Kibler, D.W. Aha, Learning representative exemplars of concepts: an initial case study, in: Proceedings of Fourth International Workshop on Machine Learning, Morgan Kaufmann, Irvine, CA, 1987, pp. 24–30.
[14] C.A.R. Hoare, Quicksort, Comput. J. 5 (1) (1962) 10–15.
[15] J.S. Aguilar, J. Riquelme, M. Toro, Three geometric approaches for representing decision rules in a supervised learning system, in: Genetic and Evolutionary Computation Conference (GECCO '99), Orlando, FL, EE.UU., 1999, p. 771.

About the Author—JOSÉ C. RIQUELME was born in Spain, 1962. He is an Associate Professor at the Technical School of Computer Science & Engineering of the University of Seville and the Chief of the department of Computer Science. He received the Ph.D. degree in Computer Science in 1996. His areas of technical interest include genetic programming, dynamical systems and data mining. http://www.lsi.us.es/∼riquelme.

About the Author—JESÚS S. AGUILAR-RUIZ was born in Spain, 1968. He is an Associate Professor at the Technical School of Computer Science & Engineering of the University of Seville. He received the Ph.D. degree in Computer Science in 2001. His areas of technical interest include evolutionary algorithms, data mining, knowledge discovery and industrial applications. http://www.lsi.us.es/∼aguilar.

About the Author—MIGUEL TORO was born in Spain, 1955. He is a Professor at the Technical School of Computer Science & Engineering of the University of Seville. He received the Ph.D. degree in Engineering in 1989. He is author of several books on dynamical systems. He is a member of IEEE and ACM.