Data Set Editing by Ordered Projection

Jesús S. Aguilar, José C. Riquelme and Miguel Toro
Departamento de Lenguajes y Sistemas Informáticos, Facultad de Informática, Universidad de Sevilla, Avda. Reina Mercedes s/n. 41012 Sevilla, Spain
E-mail: {aguilar,riquelme,mtoro}@lsi.us.es

Abstract. This paper presents a new approach to data set editing. The algorithm (EOP: Editing by Ordered Projection) has some interesting characteristics: important reduction of the number of examples from the database; lower computational cost (O($mn \log n$)) 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. The performance of EOP is analysed in two ways: percentage of reduction and classification. EOP 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 databases with continuous attributes from the UCI repository shows that EOP is a valuable preprocessing method for the later application of any axis-parallel learning algorithm.

Keywords: Data mining, preprocessing techniques, data set editing, 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 [9] is probably the most useful system 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 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 [4]. Some of them are briefly cited in the following lines. Hart [5] proposed to include in the subset S those examples of the training set T whose classification with respect to S are wrong using the nearest neighbour technique, 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. [2] proposed a variant of Hart's method; Wilson [13] 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 [11] extended the idea of Wilson eliminating the examples with incorrect classification from any $i = 1$ to k, where k is the maximum number of neighbours to be analysed; the work of Ritter [10] 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 [12] (two examples p and q are relative neighbours if for all other examples x in the set, is true the expression $\text{dist}(p, q) < \max\{\text{dist}(p, x), \text{dist}(q, x)\}$. All of these techniques need to calculate distances between examples, which is rather time consuming. If n examples with m attributes are considered, the first methods take $O(mn^2)$ time, the Ritter's algorithm is $O(mn^2 + n^3)$; the Voronoi neighbours, Gabriel neighbours and relative neighbours are $O(mn^3)$.

In this paper we present an algorithm, called EOP (Editing by Ordered Projection), which has some important characteristics:

- Considerable reduction of the number of examples.
- Lower computational cost $O(mn \log n)$ 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 [3]. To show the performance of our method we have used k-NN and C4.5 before and after applying EOP. Among the most known editing methods we have chosen IB2 [2], ENN [13] and SRHINK [7]. A 10-fold cross validation for each method is achieved to reduced 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.

2. Description of the algorithm

If we choose a region 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.

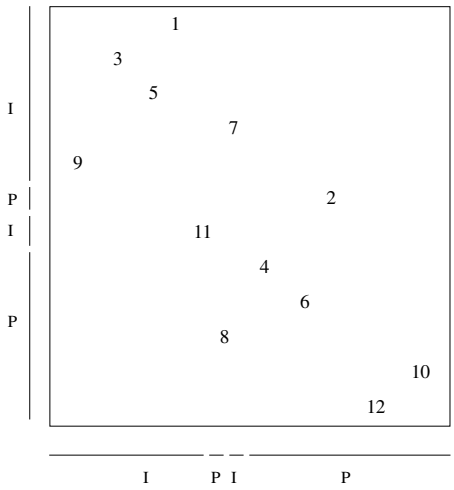

Fig. 1. An example of database.

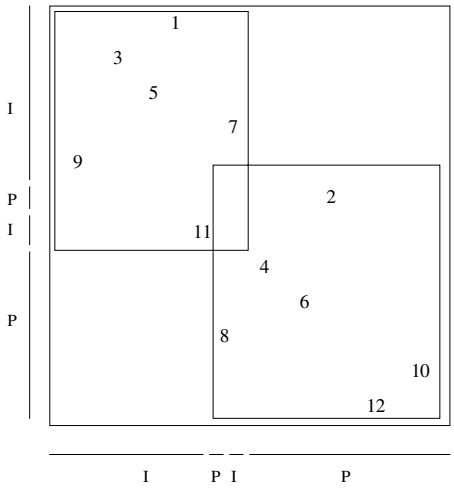

Fig. 2. The best solution with overlapped rules.

The method is completely heuristic because EOP 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 twelve numbered examples and two labels: I (odd numbers) and P (even numbers).

An optimal classifier would obtain the two rules showed 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, 4, 5 or 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 abscissas axis 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,

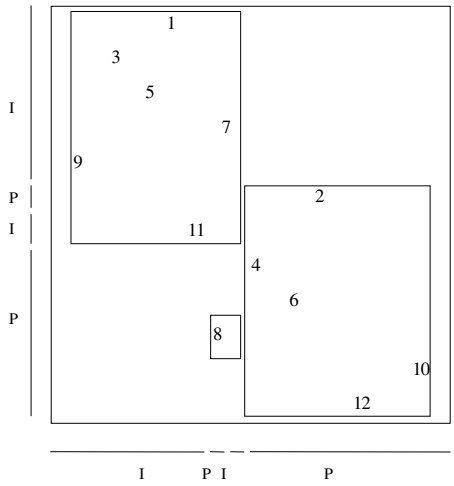

Fig. 3. One possible solution.

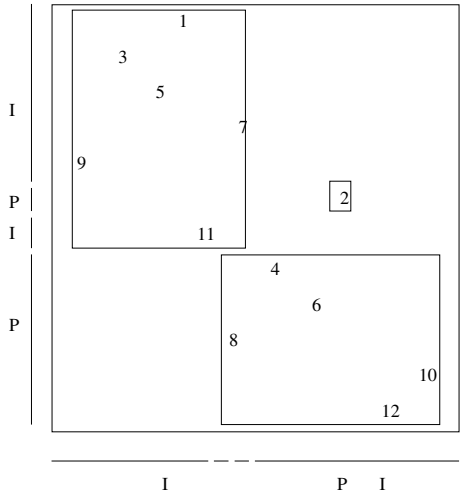

Fig. 4. One possible solution.

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 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).

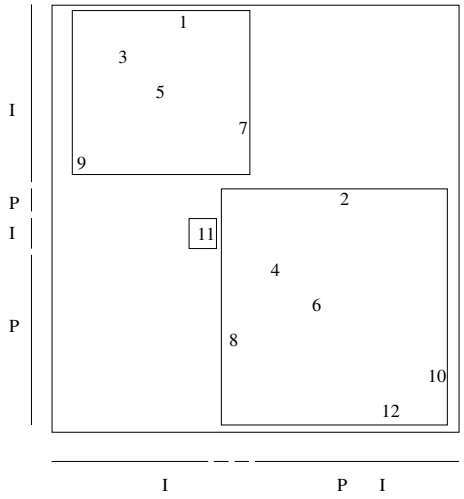

Fig. 5. One possible solution.

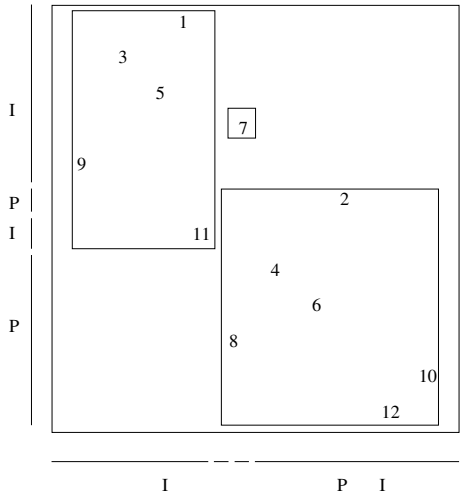

Fig. 6. One possible solution.

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 $A = I_1 \times I_2 \times \ldots \times I_m$, 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: $E \times N \rightarrow A$ and lab: $E \rightarrow C$, 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 $U =< u[1], \ldots, u[n] >$ and as the database is a sequence, the access to an example is achieved by means of its position. Likewise, the access to j-th attribute of the i-th example is made by att$(u[i], j)$, and for knowing its label lab$(u[i])$.

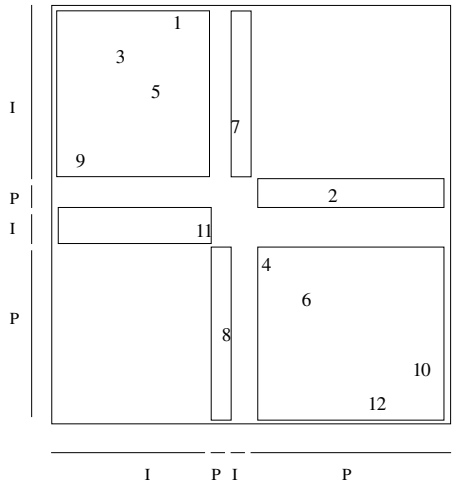

Fig. 7. Regions without overlapping.

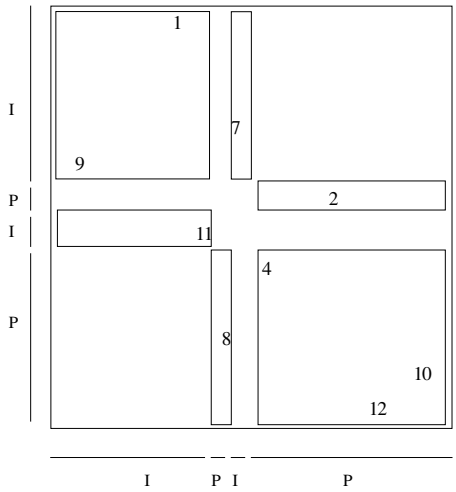

Fig. 8. Result of applying EOP.

Definition 4: An ordered projected sequence is a sequence formed by the projection of the universe onto the i-th 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,

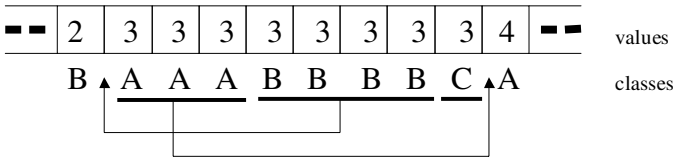

Fig. 9. QuickSort.

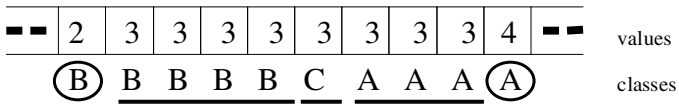

Fig. 10. ReSort.

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 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:

$$\text{weakness } = 0 \Rightarrow \text{ examples } \{9, 11, 4\}$$

$$\text{weakness } = 1 \Rightarrow \text{ examples } \{1, 8, 7, 2, 12, 10\}$$

$$\text{weakness } = 2 \Rightarrow \text{ examples } \{3, 5, 6\}$$

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. However, in some cases, it needs special treatment due to the sorting. To sort the database in increasing order by an attribute is a task achieved by the QuickSort [6] algorithm. This algorithm is $O(n \log n)$, 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 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 that 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

```
Input: E: training file (n examples, m attributes)
Output: E edited training file (n* examples)
For each example e_i ∈ E with i in {1,...,n}
    weakness(e_i) = 0
For each attribute a_j with j in {1,..., m}
    E_j = QuickSort(E_j,a_j) in increasing order
    E_j = ReSort(E_j)
    For each example e_i ∈ E_j with i in {1,...,n}
        If e_i is not border
            weakness(e_i) = weakness(e_i)+1
For each example e_i of E with i in {1,...,n}
    If weakness(e_i)=m
        remove e_i from E
```

Fig. 11. EOP algorithm.

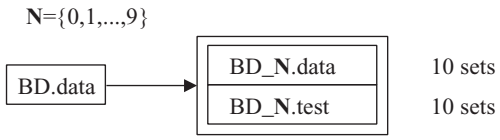

Fig. 12. Ten-fold cross-validation.

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 $O(n)$, due to the shifting of equal-valued examples. Therefore, the average computational cost of the algorithm is $O(mn \log n)$, much lower than other algorithms proposed in the bibliography, normally $O(mn^2)$.

The algorithm is illustrated in Fig. 11.

3. Experiments

Tests have been achieved in over several databases of varying complexity from the UCI repository [3]. A summary of the characteristics of these databases appears in the Appendix.

In our experiments they all use the range-normalised Euclidean Metric (EM) [14]. This function defines the distance between two values r and s of given attribute i as

$$rn\_diff_i(r,s) = \frac{|r-s|}{\max_i - \min_i}$$

The overall distance between two examples x and y is given by

$$EM(x,y) = \sqrt{\sum_{i=1}^{m} rn\_diff_i(x_i,y_i)^2}$$

For each database (BD), 10-fold cross-validation was used. A ten-fold cross-validation is performed by dividing the data into ten 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).

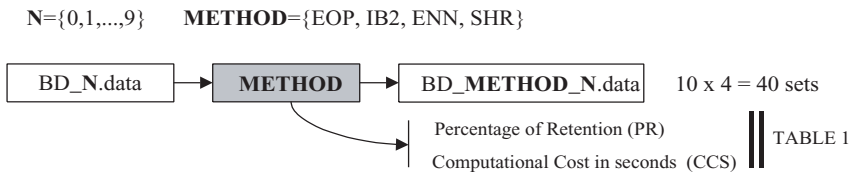

Fig. 13. Reduction methods.

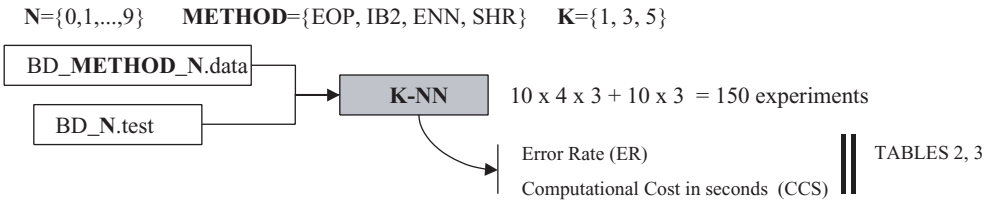

Fig. 14. Comparing the quality of reduced datasets from the editing methods by classifying with $\{1, 3, 5\}$-NN.

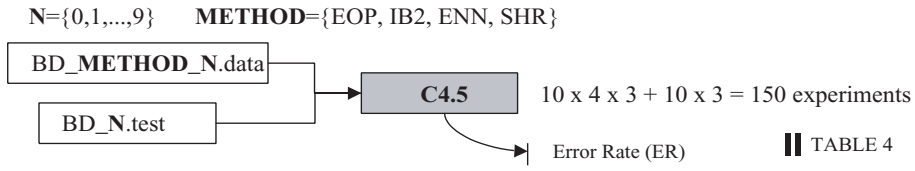

Fig. 15. Comparing the quality of reduced datasets from the editing methods by classifying with C4.5.

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), where METHOD is one of {EOP, IB2, ENN, SHR} and N is a value in $\{0, 1, \ldots, 9\}$. For example, from BD_1.data we would obtain BD_IB2_1.data by applying the IB2 method. Since we are using four methods together with ten-fold cross-validation the total amount of experiments is forty for every database.

The remainder 10% of the unseen data (BD_N.test) was tested on the instances of BD_METHOD_N.data using k-NN, with $k = 1, 3, 5$. For example, we could obtain iris_ib2_1.data by applying the IB2 method to the iris_1.data file generated by the cross validation. Afterwards, we will use iris_ib2_1.data to classify iris_1.test by means of the nearest neighbour technique varying k from one to five (odd values). We report the results from ten datasets to which four editing methods are applied and three classifications are made, so that the total amount of experiments is 120 (see $10 \times 4 \times 3$ in Fig. 14), plus the classification over the original (non-reduced) BD_N.data (see 10×3 in Fig. 14), i.e. 150.

As a further comparison, another widely-used learner, C4.5, was run on these datasets (Fig. 15). For example, after reducing iris_3.data with EOP, iris_eop_3.data was generated, it was given as input to C4.5 and the decision tree generated was used to classify the iris_3.test file (test files are never reduced). The purpose is to demonstrate that EOP is more useful than other methods if we are interested in producing axis-parallel-based models like that of C4.5 (decision trees), COGITO [1] (decision lists), and many others. EOP conserves the axis-parallel decision boundaries better than IB2, ENN and SHRINK.

The experiments show that by applying EOP, 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 ten experiments).

Table 1
Computational cost in seconds (CCS) and percentage of retention (PR) of the editing methods: EOP, IB2, ENN and SHRINK

DATABASE	EOP		IB_2		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
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
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
LETTER	331,7	34,8	4720,3	13,8	56629,8	96,1	33371,7	8,8
PIMA	2,4	74,9	8,1	38,1	45,0	71,2	32,4	37,2
average	43,3	69,9	592,2	27,2	7095,4	81,4	4182,4	26,2

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

DATABASE	KNN		EOP		IB_2		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
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
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
LETTER	6471,1	4,0	2168,2	7,0	830,8	7,9	6045,4	4,9	549,7	27,6
PIMA	4,9	28,3	3,7	30,5	2,1	35,8	3,5	24,4	1,9	39,7
average	810,7	18,2	272,2	19,6	104,3	22,7	757,2	19,9	69,1	28,2

Table 3
CCS and ER of NN, EOP, IB2, ENN and SHRINK using $\{1, 3, 5\}$-Nearest Neighbour technique

DATABASE	KNN		EOP		IB_2		ENN		SHRINK	
	CCS	ER	CCS	ER	CCS	ER	CCS	ER	CCS	ER
average 1-NN	810,7	18,2	272,2	19,6	104,3	22,7	757,2	19,9	69,1	28,2
average 2-NN	804,0	20,3	271,6	22,1	105,5	27,9	773,4	20,7	71,6	38,2
average 3-NN	807,3	22,0	274,8	22,9	106,4	29,0	772,1	22,3	70,1	39,5

The second column (PR) presents the average percentages of examples in the original training file that were included in the reduced file. As catalogued in the last row, the most salient aspect is the large difference in time between EOP 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, EOP outperforms the remainder techniques (Table 1). As the overall averages at the foot of the table indicate, with respect to the computational cost, EOP is thirteen times faster than IB2. Without taking into account letter database, the average time consumption is 2, 5 seconds for IB2 and 2.0 seconds for EOP. ENN and SHRINK are much slower, about a hundred times. SHRINK and IB2 produce the best percentages of retention (26.2% and 27.2%, respectively) at the expense of increasing the error rate.

In Table 2 the results of the classification using 1-NN are shown. The results obtained by 1-NN, 3-NN and 5-NN are very similar (see Table 3). As reflected in the average error rate, they are worsening as k is increasing, from 1 to 5. ENN and EOP reach an interesting error rate, similar to NN without reduction,

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

DATABASE	ORIGINAL	EOP	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
HAYES-ROTH	18,3	18,3	16,7	23,5	12,0
HEART	25,5	22,9	32,6	20,7	35,2
IONOSPHE	11,2	11,4	25,4	9,1	48,0
IRIS	4,0	6,0	26,0	6,0	39,3
LETTER	12,1	18,0	27,7	12,5	50,4
PIMA	26,6	28,8	31,9	25,0	36,3
average	17,2	18,7	27,1	17,0	38,9

Table 5
Global comparison

METHOD	TIME	REDUCTION	ERROR 1-NN	ERROR C4.5
1-NN	–	100.0	18.2	17.2
EOP	43.3	69.9	19.5	18.7
IB2	592.2	27.2	22.7	27.1
ENN	7095.4	81.4	19.9	17.0
SHRINK	4182.0	26.2	29.0	38.9

Table 6
Databases

Databases	#Examples	#Continuous	#Classes
Bupa liver disorder	345	6	2
Breast cancer	699	9	2
Hayes-Roth	132	5	3
Heart disease	270	13	2
Ionosphere	351	34	2
Iris	150	4	3
Letter	20000	16	26
Pima Indian diabetes	768	8	2

even though 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 (28.2%). 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 indicate that IB2 has a weaker performance when C4.5 is used. Using C4.5, EOP is more accurate than IB2 and SHRINK. 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. EOP has an error rate of 18.7, IB2, 27.1. ENN (17%) is more accurate than EOP (18.7%), but in comparison it takes a lot of time, as we observed in Table 1 (43.3 seconds versus 7095.4 seconds).

For the purposes of global comparison, in Table 5 we present the average results of the methods used in this paper. EOP has a 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 EOP.

The comparison clearly indicates that EOP 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 EOP as a preprocessing method when we are interested in proving the accuracy of a learning method based on axis-parallel representation, like decision lists or decision trees.

Conclusions

In this paper an editing algorithm (EOP: Editing by Ordered Projection) is presented. Its main application is as a preprocessing method for axis-parallel classifiers (like C4.5). EOP 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; EOP does not. The computational cost is lower than other methods $O(mn \log n)$. The test set has been realised with eight different databases from the UCI repository. The results are very interesting because they show that our algorithm is a robust method to reduce databases for studying other learning algorithms, without losing decision boundaries. EOP 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, which can help to determine the importance of an example as a decision boundary. More weakness implies less relevance. Thus, in more complicated databases we could relax the reduction factor for eliminating which weakness is greater than or equal to $m - k$ (being k an integer value in [1, m-1]), instead of m, as it is used in the algorithm above.

Appendix

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

Acknowledgements

This work has been supported by the Spanish Research Agency CICYT under grant TIC2001-1143-C03-02. Thanks to R. López de Mántaras for useful comments and suggestions.

References

[1] J.S. Aguilar, J.C. Riquelme and M. Toro, Decision queue classifier for supervised learning using rotated hyperboxes, *Lecture Notes on Artificial Intelligence* **1484** (1998), 326–336.
[2] D.W. Aha, D. Kibler and M.K. Albert, Instance-Based Learning Algorithms, *Machine Learning* **6** (1991), 37–66.
[3] C. Blake and E.K. Merz, UCI Repository of machine learning databases, 1998.
[4] T. Cover and P.Y. Hart, Nearest Neighbour Pattern Classification, *IEEE Transactions on Information Theory* **13** (1967).
[5] P.E. Hart, The Condensed Nearest Neighbour Rule, *IEEE Transactions on Information Theory* **IT-14** (1968).
[6] C.A.R. Hoare, QuickSort, *Computer Journal* **5**(1) (1962), 10–15.
[7] D. Kibler and D.W. Aha, Learning representative exemplars of concepts: an initial case study. Proceedings of Fourth International Workshop on Machine Learning, Morgan Kaufmann, Irvine, CA, pp. 24–30.
[8] V. Klee, On the complexity of d-dimensional Voronoi diagrams, *Arch. Math.* **34** (1980), 75–80.
[9] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, Publishers, San Mateo, California, 1993.
[10] G.L. Ritter, H.B. Woodruff, S.R. Lowry and T.L. Isenhour, An algorithm for a Selective Nearest Neighbour Decision Rule, *IEEE Transactions on Information Theory* **21** (1975).

[11] I. Tomek, An Experiment with the Edited Nearest-Neighbour Rule, *IEEE Transactions on Systems, Man, an Cybernetics* **SMC-6** (1976).
[12] G.T. Toussaint, The relative neighbourhood graph of a finite planar set, *Pattern Recognition* **12**(4) (1980), 261–268.
[13] D. Wilson, Asymptotic Properties of Nearest Neighbour Rules using Edited Data, *IEEE Transactions on Systems, Man and Cybernetics* **2** (1972).
[14] R. Wilson, Advances in Instance-Based Learning Algorithms, Doctoral Dissertation, Brigham Young University, 1997.