Impact of learning set quality and size on decision tree performances

IMPACT OF LEARNING SET QUALITY AND SIZE ON DECISION TREE PERFORMANCES

M. Sebban[1], R. Nock[2], J.H. Chauchat[3] and R. Rakotomalala[3]

[1]TRIVIA, Dept of Mathematics and Computer Science
UFR Sciences, Campus Fouillole, French West Indies and Guiana University
97159 Pointe-A-Pitre Cedex, France

[2]TRIVIA, Interfaculties Scientific Dept
UFR Sciences, Campus of Schoelcher, French West Indies and Guiana University
97278 Schoelcher Cedex, France

[3]ERIC-Lyon2 Laboratory
Dept of Economics, University Lyon 2, 69676 Bron Cedex, France

e-mail: msebban@univ-ag.fr,rnock@martinique.univ-ag.fr
{chauchat,rakotoma}@univ-lyon2.fr

Accepted in the final form: 15 November 2000

Abstract. The quality of a decision tree is usually evaluated through its *complexity* and its *generalization accuracy*. Tree-simplification procedures aim at optimizing these two performance criteria. Among them, *data reduction techniques* differ from *pruning* by their simplification strategy. Actually, while pruning algorithms directly control tree size to combat the overfitting problem, data reduction techniques perform a data preprocessing prior to decision tree construction to improve the learning set quality. Recent experimental results have shown that randomly manipulating training set size has a direct impact on tree size, and therefore recommend the use of the latter simplification strategy. In this paper, we provide theoretical arguments justifying data preprocessing in favor of tree simplification. We also investigate new data reduction techniques, usually used in the field of *prototype selection.* From experiments with 22 datasets, we show that some of them are very efficient to improve standard post-pruning performances.

Keywords: decision tree, tree simplification, prototype selection, pruning.

1 Introduction

Most models used in machine learning require a learning stage on a training set. The relevance and the number of examples in this training sample widely influence the future performances of the induced model. For instance, *overfitting* is an observed pathology of decision trees, one of the most popular

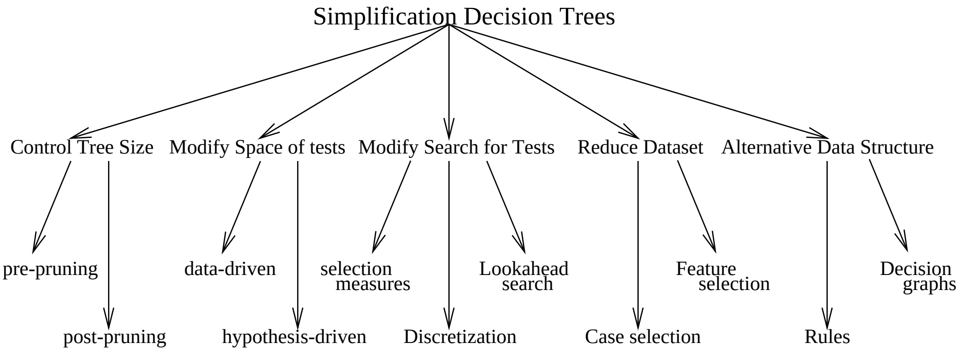

Figure 1: Framework for Tree-Simplification Procedures.

approach in the Knowledge Discovery in Databases field. The induction algorithm may actually overfitt *outliers*, *mislabeled*, *noisy data* resulting in the inference of more structures than is justified by the training set. Thus, decision trees (and the induced knowledge) are more complex and often incomprehensible, even by an expert of the studied domain. In such a context, many methods for simplifying trees have been proposed in literature during the last decades. Breslow and Aha [5] have proposed a clustering of such methods in machine learning according to their simplification strategies (figure 1).

Among them, *pruning* techniques are probably the most popular, as a way to *a posteriori* correct overfitting. Most pruning algorithms perform a post-order traversal of the tree, examine individual subtrees and remove those subtrees deemed to be irrelevant according to a given estimate of error: *error-based pruning* [19], *reduced-error* [18], *minimum description length* [20], *cost-complexity pruning* [4], *loss-based pruning* [3], etc. By removing unnecessary subtrees, pruning techniques locally "forget", from a given level of the tree, some instances judged difficult to learn according to a given criterion.

A second strategy for simplifying decision trees consists in *a priori* reducing the size of the original learning set (called *data reduction techniques,* the fourth category in figure 1). This aim can be achieved by removing *irrelevant features* before the tree induction process, often resulting in smaller trees. A study in [24] shows that such a strategy can be very efficient. Using C4.5 [19], the non deletion of weakly relevant features generates actually deeper decision trees with lower performances. *Ad hoc* algorithms have been proposed to deal with this phenomenon on decision trees [2, 8]. A second way to proceed consists in removing *irrelevant training instances* prior to tree construction. Such algorithms have been presented during the last decades in [17] (windowing strategy), [25] (sampling strategy) and [11] (with John's *Robust C4.5* decision trees).

In this paper, we only focus on these *case selection techniques.* Following the example of the standard pruning, such methods have an effect on the *decision tree size* and the *accuracy generalization*, which are the two main performance measures to optimize. However, their difference comes from the fact that removed instances are definitely eliminated prior to tree construction. From a practical point of view, three main arguments justify such data preprocessing instead of a standard pruning:

1. Firstly, pruning is statistically dependent on local distributions in each node of the tree. In some cases, pruning is not performed because of the presence of irrelevant instances, resulting in the keeping of the node split. Figure 2 presents a 2D-simulated example. $C4.5$ [19] is run on the

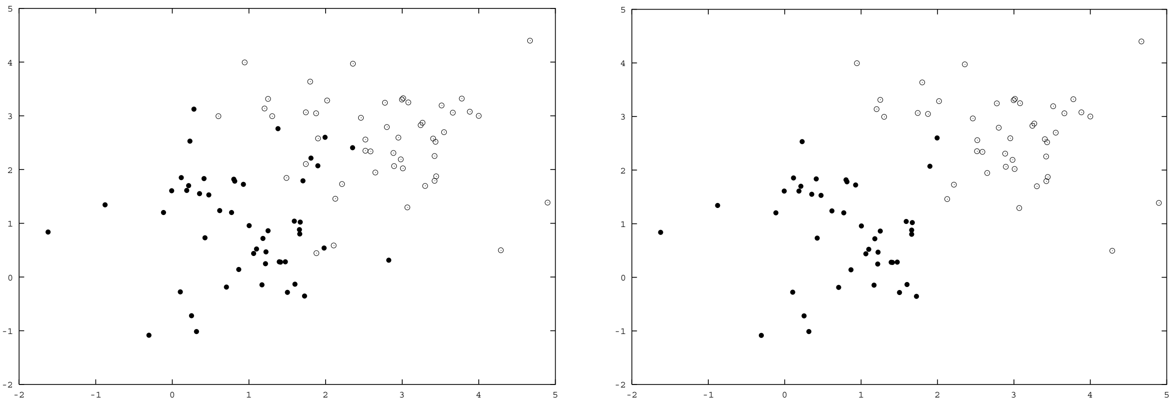

Figure 2: The left-hand sample is the original learning set (two overlapped gaussian laws); the right-hand sample is the deduced set after removing mislabeled instances by John's *RC4.5*.

original learning set (on the left) and on the dataset reduced by the John's Robust $C4.5$ (on the right) [11]. Using the whole learning set, the model overfitts the data (figure 3). Starting with the reduced data set, the decision tree is smaller and more accurate.

2. Secondly, under some circumstances, the additional cost of the data preprocessing (DP) is smaller than the additional cost due to the calculations on the whole learning set. Thus, we have often experimentally noted on noisy data that $time(\text{DP}+C4.5_{subset}) < time(C4.5_{whole\ set})$, even if this claim can not be theoretically justified and expressed from a complexity point of view, because it depends on the presence of noise. Of course, on unnoisy datasets a data preprocessing is often useless, and computationally expensive.

3. The last argument in favor of a data preprocessing comes from recent empirical results [16] which show that manipulating training set size will have a direct impact on tree size. Said differently, authors claim that an important part of the tree size reduction is simply due to the reduction of the learning sample. That means that even if $C4.5$ (or another induction algorithm) is pruning unnecessary subtrees, overfitting is in fact occurring and its gets worse as the size of the learning set increases. Experimental results presented in [16] with five pruning methods seem to confirm this hypothesis. By randomly adding new instances to the learning set, decision tree size grows even after the accuracy has ceased to increase.

While these recent works seem to empirically justify the use of case selection techniques, we provide in this paper theoretical arguments justifying data preprocessing instead of performing a pruning on the whole learning set. Actually, we statistically prove that the probability to perform an *error-based pruning* [19] converges on 0 when the learning set size grows. Then, if our main objective consists in reducing decision tree sizes, it is in our interest to reduce the learning set size before the construction of the tree. In [16] authors have empirically shown that a random reduction is actually very efficient for reducing decision trees, while not compromising the generalization accuracy. We claim in this paper that efficient case selection techniques not only would reduce tree size, but also would improve the generalization accuracy of the induced model. To validate this hypothesis, we

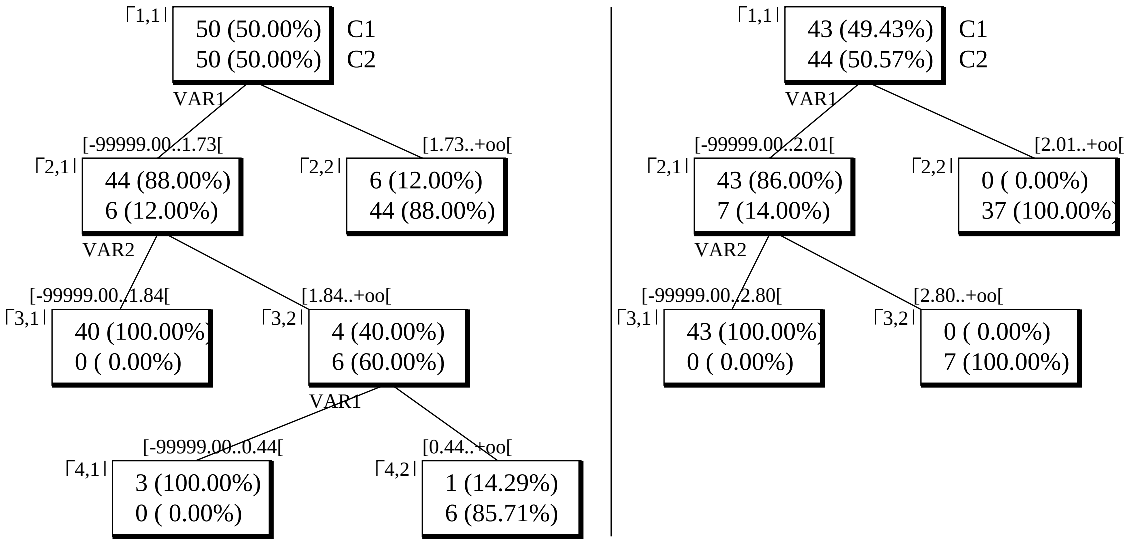

Figure 3: Two decision trees built with the standard $C4.5$ before and after the *Robust* $C4.5$ data preprocessing: on the left, using the whole learning set, the model overfits the data. Starting with the reduced data set, the decision tree is smaller and more accurate.

investigate new case selection techniques, not originally proposed for reducing tree size and improving human comprehensibility, but rather for limiting database storage requirements in memory [6, 9, 10, 15, 22, 26]. These algorithms (often called prototype selection methods) have been developed in order to compensate for the drawback of case-based learning algorithms, not only well known to be very efficient, but also computationally expensive. As far as we know, no recent study has attempted to establish a close link between the best prototype selection procedures and tree simplification. This behavior can be in fact doubly justified:

- The first reason comes from the main goal of the prototype selection algorithms: they are commonly used to reduce storage requirements imposed by expensive algorithms such as the k Nearest-Neighbors (kNN). Therefore, their reduction strategies are not *a priori* directed towards the construction of smaller trees.

- The second reason directly ensues from the first. Since most of these algorithms usually use a kNN classifier during the selection, their adaptation to tree simplification is *a priori* not natural.

Nevertheless, despite this established fact, we show in this paper that some of prototype selection methods constitute good candidates for simplifying trees, in comparison with the standard pruning, and the state-of-the-art tree simplification methods. The rest of this paper is organized as follows:

In section 2, we review the tree simplification via data reduction techniques, and we present recent results on the relationship between tree size and training set size. Section 3 introduces our theoretical results about the effect of training set size on $C4.5$ pruning. In section 4, we review the prototype selection techniques, and provide some of pseudocodes of these reduction algorithms. Finally, in section 5, we present a large comparative study of selected algorithms on 22 datasets, before giving a conclusion.

2 Tree Simplification via Data Reduction Techniques

One can tackle the problem of tree simplification via data reduction techniques, either through *feature* selection or *case* selection. Both strategies have shown efficient results, improving the quality of the learning sample. Even if we only focus in this paper on case selection methods, we describe in this section a brief review concerning the two approaches. We also present recent results relating to relationships between training set size and tree size.

2.1 Tree Simplification via Feature Selection

To achieve feature selection, one usually uses one of the two following approaches, called *wrapper* and *filter* models [12]. While wrapper models assess alternative feature subsets using a given classification algorithm (which does not always use a decision tree representation), filter models use a preprocessing stage, before the induction process, to select relevant features. The parameter to optimize is often a statistical criterion or an information measure [13, 21]. Tree simplification methods via feature selection use an induction algorithm and optimize the generalization accuracy. Then, they belong to wrapper models. For instance, in [8] authors introduce SET-Gen, an algorithm that improves the comprehensibility of decision trees grown by standard C4.5 without reducing accuracy. It uses genetic search to select the set of relevant features. In [12], authors describe a method for feature subset selection using cross-validation that is applicable to any induction algorithm ($C4.5$ [19], $CART$ [4], $ID3$ [17], etc.).

2.2 Tree Simplification via Case Selection

Case selection techniques reduce the number of training instances in order to reach the following explicit goal: improving the quality of the learning sample in order to simplify trees and at least control generalization accuracy of the model. In such a context, many simplification strategies can be performed. Quinlan's strategy [17] consists in incrementally adding misclassified instances to an original learning sample and inducing a new decision tree until no cases are misclassified. To avoid to completely re-induce the tree, that is computationally expensive, Utgoff [25] updates trees incrementally as each new case is added. John's Robust-C4.5 [11] chooses an opposite strategy by iteratively removing all instances misclassified by the current decision tree and building a new one. Note that the tree is re-induced at each stage in the original algorithm, even if John claims that *RC4.5* could be modified with a scheme such as that proposed in [25]. Since we will use this simplification method in our experiments, we describe in figure 4 the pseudocode of $RC4.5$.

```
ROBUSTC45(TrainingData)
 Repeat
    $T \leftarrow C45BuildTree(TrainingData)$
    $T \leftarrow C45PruneTree(T)$
    foreach Record in TrainingData
       if $T$ misclassified Record then
       remove record from TrainingData
 Until T correctly classifies all Records in TrainingData
```

Figure 4: Pseudocode of the $ROBUST$-$C4.5$ algorithm

Brodley and Friedl [6] have proposed a new approach to identifying and eliminating mislabeled training instances. Then, their aim is not really the reduction of the database, but rather the improvement of the quality of the learning data. They introduce the concept of *Consensus Filter*. This approach constructs a set of base-level detectors (*i.e.* classifiers) and uses them to identify mislabeled instances by consensus vote, *i.e.* all classifiers must agree to eliminate an instance. The base-level detectors can come from different fields (*neural network, decision tree, kNN, genetic algorithm*, etc.). The pseudocode of this method (called CF) is described in figure 5. If one only uses kNN classifiers (for different values of k) as base-level detectors, this algorithm becomes very suited to reducing computational costs and improving performances of case-based learning algorithms, such as the nearest-neighbor classifiers [22]. Brodley and Friedl also noted that applying the consensus filter to the training data leads to *substantially smaller decision trees*. They achieved this aim using the following base-level detectors: a 1-NN, a linear machine and an univariate decision tree. Moreover, we can note that if CF uses only $C4.5$ as base-level detector, it corresponds in fact to $RC4.5$ with only one iteration.

In order to assess the consensus filter's ability to identify mislabeled instances, authors also provided estimates of the probability of throwing out *good* data and the probability of keeping *bad* data, easily computable on simulated noisy datasets. We will explain in detail and use these estimates in the experimental section.

```
CF(TrainingData)
 For Each Base-Level Detectors BLD$_i$
    $C_i \leftarrow BLD_i(TrainingData)$
 End For
 Foreach Record in TrainingData
    if all $C_i$ misclassified Record then
    remove record from TrainingData
 End For
 Return the TrainingData
```

Figure 5: Pseudocode of the CF algorithm

2.3 Linear Relationship Between Training Set Size and Tree Size

While previous works have proposed case reduction methods for simplifying trees and controlling generalization accuracy, other works have tried during the last decade to establish relationships between training set size and tree size. Such relationship was originally identified by Catlett [7], who found that decision trees built from a part of large learning samples were smaller than those constructed on the whole datasets, even if the accuracy was not always controlled.

An important advancement was achieved by Oates and Jensen [16]. Authors have tested the hypothesis that there is a nearly linear relationship between training set size and tree size, even after accuracy has ceased to increase. The experiments were based on a k-cross validation procedure, retaining randomly from 5% to 100% of the original instances. For each subset of instances, 5 pruning methods were applied, generating tree size and accuracy curves. Results of the linear regression of tree size on training set size provide high values of R^2 particularly for the error-based pruning [19], that seems to confirm their hypothesis. We will explain from a statistical point of view in the next section why $C4.5$ is particularly sensitive to the increase of training set size.

Authors also proposed an experimental method for evaluating new data reduction methods. The impact of a given data reduction technique must be assessed through two components: *the part due to the reduction of the training set size, and that which is directly attributable to the selection strategy.* To deal with this problem, Oates and Jensen defined four items of information:

- the size of the tree that $C4.5$ builds on the whole learning set (LS), called *C4.5 Size*,
- the size of the tree built on the subset *SB1* of instances selected by the data reduction technique, called *C4.5 DRT Size,*
- the percentage of learning instances retained by the data reduction technique, called *%Kept*,
- the size of the tree built on the subset *SB2* containing *%Kept* of instances randomly selected, called *C4.5 Random Size*

Authors propose the following criterion to estimate the effect on tree which is due to the reduction in the training set size:

$$Effect_{LS\ size/Tree\ size} = 100 * \frac{(C4.5\ Size - C4.5\ Random\ Size)}{(C4.5\ Size - C4.5\ DRT\ Size)}$$

A value of $Effect_{LS\ size/Tree\ size}$ near 0 means that reduction of training set size accounts for 0% of the effect of the data reduction technique. Said differently, it means that all the tree reduction is simply due to the relevance of the selected instances, and not to the reduction of the learning sample. On the contrary, a value of $Effect_{LS\ size/Tree\ size}$ near 100% means that the tree reduction is attributable to the fact that the learning sample is reduced. Finally, a value near 50% means that the tree reduction is as much due to the learning set reduction as the relevance of the selected instances. Authors have performed this decomposition for the $RC4.5$ algorithm, and shown that an important percentage of its effect on tree size is attributable to the fact that it simply reduces the size of the training set. On 12 benchmarks, they show that 42% of $RC4.5$'s effect is due to reduction of training set size.

In fact, we can claim that such criterion for evaluating $RC4.5$'s effect is relevant when the accuracy computed from *SB1* is the same as for the subset *SB2*. In such a context, it is actually possible to directly compare tree sizes and evaluate $RC4.5$'s contribution. If this constraint is not satisfied, the relevance of the reduction method can only be assessed through the analyze of two performance measures: tree size, and generalization accuracy. We will analyze this problem in the experimental section.

3 Effect of training set size on C4.5 pruning

3.1 Introduction

The first tangible results about the relationship between training set size and tree size have been presented in [16]. Authors have empirically shown that, under a broad range of circumstance, all data reduction techniques (especially $C4.5$ in their experiments) result in some decrease in tree size with little impact on accuracy. In this section, we provide theoretical justifications of this trend for $C4.5$ in the case with 2 classes (the principle remains the same for multiclass problems).

Consider a given node 0 of the decision tree, split into two nodes 1 and 2. Our goal consists in evaluating the pruning probability of this subtree when the training set size increases. The learning set is supposed to be produced by a simple random sampling process from the reference population Ω, that gives the same probability to each possible sample set.

3.2 Theoretical Risk Behavior in Subtrees

Let π_{ij} be the relative proportion of instances of the class i in the node j **in the whole population** Ω. Let $\theta_0 = MIN(\pi_{1.};\pi_{2.})/\pi_{..}$ be the error rate in the node 0 and $\theta_1 = MIN(\pi_{11};\pi_{21})/\pi_{.1}$ and $\theta_2 = MIN(\pi_{12};\pi_{22})/\pi_{.2}$ respectively the error rates in the nodes 1 and 2 in Ω. Without loss of generalization, we assume here that $\pi_{1.} > \pi_{2.}$, that means that class 1 has the majority in the node 0, therefore $\theta_0 = \pi_{2.}/\pi_{..}$. The average error rate from the subsequent nodes is:

$$\theta = \theta_1 \times \ \pi_{.1}/\pi_{..} + \ \theta_2 \times \pi_{.2}/\pi_{..} = \left[MIN(\pi_{11};\pi_{21}) + MIN(\pi_{12};\pi_{22})\right]/\pi_{..}$$

There are four possible configurations for the error rates in the subsequent nodes:

	MIN(π_{12};π_{22})=π_{12}	MIN(π_{12};π_{22})=π_{22}
MIN(π_{11};π_{21})=π_{11}	Impossible ($\pi_{1.} > \pi_{2.}$ is assumed)	$\pi_{..} \times \theta$=π_{11}+$\pi_{22} < \pi_{12}$+π_{21}=$\pi_{2.}$
MIN(π_{11};π_{21})=π_{21}	$\pi_{..} \times \theta$=π_{12}+$\pi_{21} < \pi_{21}$+π_{22}=$\pi_{2.}$	$\pi_{..} \times \theta$=π_{21}+π_{22}=$\pi_{2.}$

Therefore, the average error rate θ is always smaller or equal to θ_0. Said differently, either majority classes are different in nodes 1 and 2 resulting in the decrease of θ_0, or the majority class is the same resulting in the stabilization of θ_0 (this claim is true in the whole population and in the sample set).

3.3 Pruning Probability

Let N be the size of the **learning sample** LS and N_{ij} be the number of examples of the class i observed in the node j. Let p_j be the empirical risk in the node j.

In $C4.5$ algorithm, a subtree is pruned if and only if:

$$\begin{gathered}\left[\tfrac{N.1}{N}(p_1 + 1.96\sqrt{\tfrac{p1\times(1-p1)}{N.1}}) + \tfrac{N.2}{N}(p_2 + 1.96\sqrt{\tfrac{p2\times(1-p2)}{N.2}})\right] > (p_0 + 1.96\sqrt{\tfrac{p0\times(1-p0)}{N}}) \\ \Longleftrightarrow \left(\tfrac{N.1}{N}p_1 + \tfrac{N.2}{N}p2 - p_0\right) + 1.96\left(\tfrac{N.1}{N}\sqrt{\tfrac{p1\times(1-p1)}{N.1}}) + \tfrac{N.2}{N}\sqrt{\tfrac{p2\times(1-p2)}{N.2}} - \sqrt{\tfrac{p0\times(1-p0)}{N}}\right) > 0 \qquad (1)\end{gathered}$$

The second term in brackets tends to 0 as N tend to ∞, because it can be seen as a bounded term divided by $\sqrt{N}$. Actually,

$$\begin{gathered}\tfrac{N.1}{N}\sqrt{\tfrac{p1\times(1-p1)}{N.1}} + \tfrac{N.2}{N}\sqrt{\tfrac{p2\times(1-p2)}{N.2}} - \sqrt{\tfrac{p0\times(1-p0)}{N}} = \\ \tfrac{1}{\sqrt{N}}\left[\sqrt{\tfrac{N.1}{N}}\sqrt{p1\times(1-p1)} + \sqrt{\tfrac{N.2}{N}}\sqrt{p2\times(1-p2)} - \sqrt{p0\times(1-p0)}\right]\end{gathered}$$

Then the sign of the expression (1) depends on the value of the first term $(\frac{N.1}{N}p_1 + \frac{N.2}{N}p2 - p_0)$. Two different situations can occur:

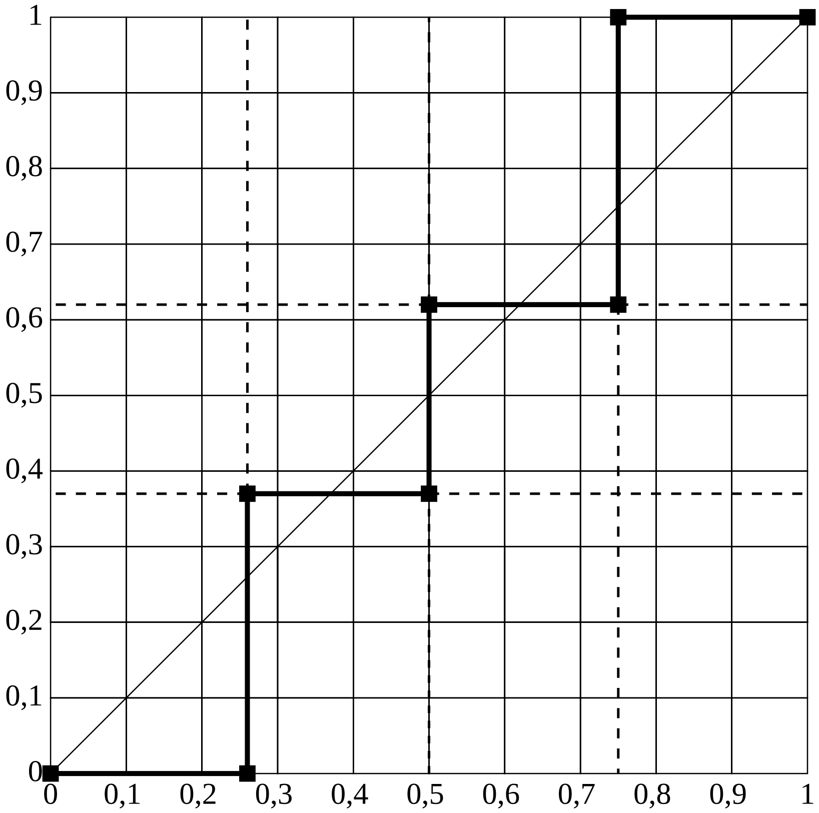

Figure 6: Progressive adjustment of the decision rule built by C4.5 on a linearly separable problem.

1. First situation (the most common): in the reference population Ω, the majority classes are different in nodes 1 and 2, then the average error rate is decreasing and $\left(\frac{N.1}{N}p1 + \frac{N.2}{N}p2 - p0\right)$ tends towards its expected value $(\pi_{.1} \times \pi_{21} + \pi_{.2} \times \pi_{12} - \pi_{2.})$ or $(\pi_{.1} \times \pi_{11} + \pi_{.2} \times \pi_{22} - \pi_{2.})$, both of them < 0. In this case the pruning probability tends towards 0 as N tends towards ∞; *then the tree size built by C4.5 increases with the learning set size N.*

2. Second situation: in Ω, the majority class is the same in nodes 0, 1 and 2, then the average error rate is stable and $\left(\frac{N.1}{N}p1 + \frac{N.2}{N}p2 - p0\right)$ oscillates around its expected value 0. In fact this situation is rare because the nodes tends to specialize, especially around the optimum frontier between classes in Ω, resulting in the growing of the tree.

3.4 Example

Consider a problem with 2 classes, where each example is represented by a 2-dimensional vector ($X_1 \in [0-1]$, $X_2 \in [0-1]$). In the population Ω, the density is uniform in this 1×1 square. The example belongs to the class 1 if $0 < X_1 < X_2 < 1$ and to the class 2 if $0 < X_2 < X_1 < 1$. Each class has an uniform density in its half-square. A theoretical study of the C4.5's *gain-ratio* function (see figure 6) shows that the first optimal cutting takes place at $X_1 = 0.5$, with $X_2 = 0.374$ if $X_1 < 0.5$

and at $X_2 = 0.626$ if $X_1 > 0.5$. Proceeding this way, the tree built by $C4.5$ adjusts progressively the optimum diagonal frontier by vertical and horizontal line segments as shown in figure 6. The generalization accuracy on a training set will converge to its theoretical limit (which is here equal to 1), as N tends to ∞.

Even if this example seems to be very simple, several real world problems have, at least locally, linear theoretical frontiers. According to this theoretical problem, we generated several learning sets with N varying from 100 to $200,000$. In order to reduce the variance, we generated 10 learning samples for each size N, and we used the average results. For each dataset, we evaluate the tree complexity built by $C4.5$ (the number of leaves) and its generalization accuracy computed from a test set with $500,000$ examples (see figure 7). As expected, we can note that tree size increases with N and continues to grow even after accuracy has ceased to increase and reached its theoretical limit (here equal to 1).

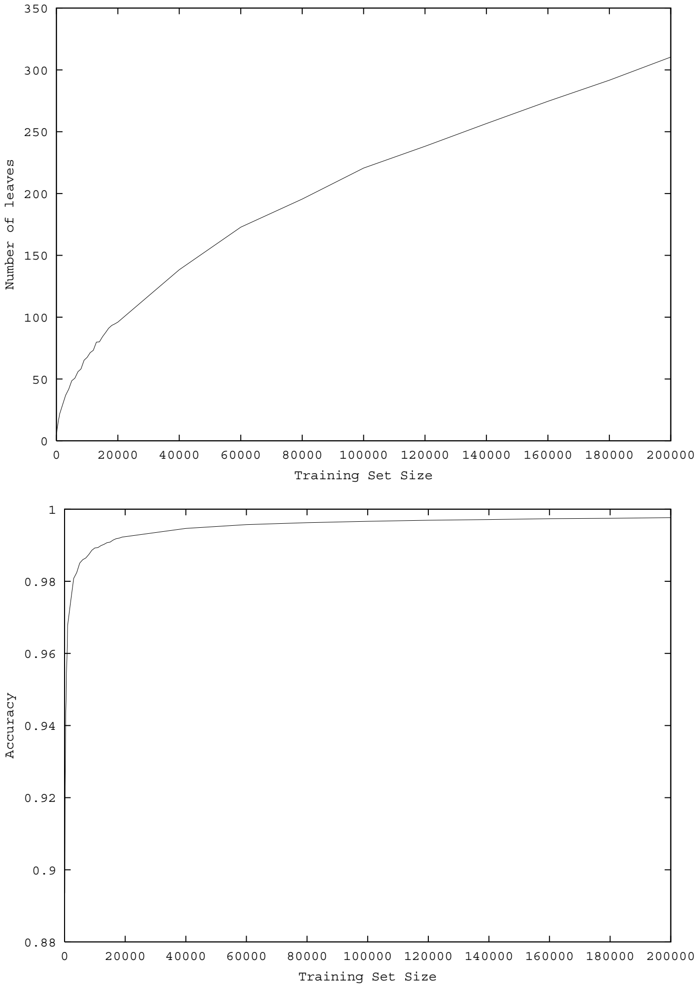

Figure 7: Tree size and accuracy as a function of training set size.

All these results confirm the relationship empirically established in [16] between tree size and training set size. A possible interpretation of our theoretical result would consist in reversing the problem, *i.e.*:

1. if the training set size decreases by *randomly* eliminating instances, then the tree size built by $C4.5$ also decreases, while controlling the generalization accuracy.

2. according to 1., we can submit the following hypothesis: if the training set size decreases by *strategically* eliminating irrelevant instances, then not only the tree size built by $C4.5$ decreases, but also the generalization accuracy can be improved.

The consequence of this hypothesis is simple: a data preprocessing consisting in removing irrelevant instances prior to decision tree construction would be more efficient (from a complexity and performance point of view) than a standard pruning.

Results presented in [6, 11] seem to confirm this hypothesis. Oates and Jensen [16] claim that a part of the tree size reduction is simply due to the reduction of training set size, and the rest is due to the strategy used for selecting relevant instances. In such a context, the aim of the following section consists in testing our hypothesis and investigating new data reduction techniques, usually called *prototype selection* methods, as a data preprocessing.

4 Prototype Selection Techniques

Historically, prototype selection (PS) firstly aimed at improving the efficiency of the nearest-neighbor classifier [10]. While its use was widely spread and encouraged by early theoretical results linking its generalization error to Bayes, this classifier presents several problems from a practical point of view [4]:

1. It is a computationally expensive classifier because it stores all the instances in memory,

2. It is intolerant of noisy instances,

3. It is intolerant of irrelevant attributes,

4. It is sensitive to the chosen distance function,

Pioneer works in PS firstly searched only solutions to solve the first problem listed above. In [10], Hart proposes a *Condensed NN Rule* (CNN) to find a *Consistent Subset*, *CS*, which correctly classifies all of the remaining points in the sample set. However, this algorithm does not find a *Minimal Consistent Subset, MCS.* The *Reduced NN Rule* (RNN) proposed by Gates [9] tries to overcome this drawback. It searches for the minimal subset in Hart's *CS* which correctly classifies all the learning instances. This approach is efficient if and only if Hart's *CS* contains the *MCS* of the learning set, and that is not always the case. In [1], the IB2 algorithm is quite similar to the CNN rule, except it does not repeat the process after the first pass. A common characteristic of these approaches is that they are very sensitive to noise because noisy instances will probably be misclassified, and then not deleted. Initial solutions to improve IB2, and then handle the second problem listed above, were proposed in IB3 [1].

The third problem (intolerance of irrelevant attributes) is a matter of *feature selection.* Therefore, we do not discuss it in this section. Concerning the choice of the distance function, PS algorithms usually use the Euclidean distance. This function is appropriate only if the attributes are numeric. In [27], Wilson and Martinez propose new distance functions to handle numeric and nominal attributes.

In [23], Skalak proposes two algorithms to find sets of prototypes for NN classification. The first one is a Monte Carlo sampling algorithm, and the second applies random mutation hill climbing. In these two algorithms, the size of the prototype subset must unfortunately be provided by the user. Moreover, they require the fixing of another parameter which contributes to the increase in the time complexity: the number of samples or the number of mutations. All these user-fixed parameters make the algorithms too user-dependent, despite interesting performances.

```
RT3(Training Set LS)
  Let S=LS
  Noise filtering
  Sort instances in S by the distance to their nearest ennemy
  For Each instance P in S
    Find $P.N_{1..k+1}$, the $k+1$ nearest neighbors of P in S
    Add P to each of its neighbors' lists of associates
  For each instance P in S:
    Let with=# of associates of P classified correctly with P as a neighbor
    Let without=# of associates of P classified correctly without P
    If $(without - with) \geq 0$
      Remove P from S
      For each associate A of P
        Remove P from A's list of nearest neighbor
        Find a new nearest neighbor for A
        Add A to its new neighbor's list of associates
    End If
  Return S
```

Figure 8: Pseudocode of $RT3$

In [26], Wilson and Martinez present the algorithm *RT3* (it proceeds as shown in Figure 8), probably one of the most efficient PS algorithm. In this approach, an instance ω_i is removed if its removal does not hurt the classification of the instances remaining in the sample set, notably instances that have ω_i in their neighborhood (called *associates*). The algorithm is based on 2 principles: it uses a noise-filtering pass, which removes instances misclassified by their kNN, that avoids overfitting the data; it removes instances in the center of clusters before border points, by sorting instances according to the distance from their nearest *enemy* (*i.e.* of a different class).

In Sebban and Nock's $PSRCG$ [22], authors investigate the prototype selection as an information preserving problem (see the pseudocode of $PSRCG$ in Figure 9). Rather than optimizing the accuracy of a classifier, they build a statistical information criterion (RCG in the algorithm) based on a quadratic entropy computed from the nearest-neighbor topology. From neighbors linked by an edge to a given instance, $PSRCG$ computes a quadratic entropy by taking into account the label of each neighbor. From this entropy (which conveys a local uncertainty U_{loc}), it deduces a global uncertainty U_{tot} of the learning sample LS. While an instance deletion is statistically significant, $PSRCG$ eliminates uninformative examples. Using $k-$nearest neighbors for building the neighborhood graph, $PSRCG$ has shown a high ability for reducing the database and improving kNN performances [22].

```
PSRCG ALGORITHM
 $t \leftarrow 0$; $N_p = |LS|$
 Build the kNN graph on the learning set
 Compute $RCG_1$
               $RCG_1 = \frac{U_0 - U_{tot}}{U_0}$
 Repeat
    $t \leftarrow t + 1$
    Select the instance $\omega$ = max($U_{loc}(\omega_j)$)
    If 2 instances have the same $U_{loc}$
    select the example having the smallest
    number of neighbors
    Local modifications of the kNN graph
    Compute $RCG_{t+1}$ after removing $\omega$
    $N_p \leftarrow N_p - 1$
 Until ($RCG_{t+1} < RCG_t$) or not($RCG_{t+1} >> 0$)
 Remove instances having a null uncertainty with their ($k + 1$)-NN
 Return the prototype set with $N_p$ instances
```

Figure 9: Pseudocode of *PSRCG*

Finally, the *PSB* algorithm [15] proposes an adaptation of the properties of boosting to prototype selection. While in a standard boosting algorithm the final classifier combines a set of weak hypotheses, where each one is a classifier built according to a given distribution over the training data, each weak hypothesis in *PSB* is a single weighted prototype. The distribution update (a key step of boosting) and the criterion optimized during the process are slightly modified to allow an efficient adaptation of boosting to the prototype selection field.

5 Tree Simplification via Prototype Selection: a Comparative Study

5.1 Introduction

In this section, we achieved an important programming work in order to evaluate the contribution of prototype selection algorithms to tree simplification. To achieve this aim, we compared them not only with the state-of-the-art case selection methods but also with the standard *error-based pruning* [19]. We updated the SIPINA software, developed in the ERIC-Lyon2 laboratory[1] and the GRAF software developed in the TRIVIA laboratory[2]. Deliberately, each PS algorithm has been applied using its original specificities (*i.e.* to reduce the storage requirement), without any methodological adaptation to tree simplification.

Therefore, *RNN* and *RT*3 were run using a kNN classifier (here $k = 5$) during the selection (note that Hart's *CNN* is not tested here because it contains the subset deduced by Gates's *RNN rule*). *PSRCG* has been performed from a *5*-NN graph. *PSB* requiring to provide the number N_w of weak hypothesis, we decided to fix N_w as the number of prototypes selected by *PSRCG*. Such a strategy allows comparisons between approaches.

[1] http://eric.univ-lyon2.fr

[2] http://www.univ-ag.fr

The problem for the Consensus Filter (CF) was more complicated, because it requires to *a priori* provide not only the number of base-level detectors but also their type. For this experimental study, we decided (as it was performed in [6]) to take three classifiers: a *1*-NN classifier, *C4.5,* and a linear machine. These three classifiers must agree to remove an instance with CF. Finally, John's $RC4.5$ was also run.

Each data reduction method was applied as a data preprocessing before the construction of the decision tree by the $C4.5$ algorithm. We tested here a large panel of databases (22 datasets), the large majority coming from the UCI Repository [14]. The experimental method was the following: f-fold cross-validation (here $f = 5$) was used on each database to obtain unbiased estimates of the true performance of $C4.5$ algorithm according to the data reduction method. Each database DB is then divided into f disjoints sets DB_i. Data preprocessing (successively RNN, $RT3$, $PSRCG$, PSB, CF and $RC4.5$) is applied on each combination $DB - DB_i$. A tree is built with $C4.5$ on the resulting subset of instances $(DB - DB_i)_{subset}$ and tested on the instances in DB_i. For each data reduction method, we obtain an accuracy estimate by averaging results over all f folds. Note that $C4.5$ is also run for comparisons on each combination $DB - DB_i$ without data preprocessing and the induced tree is tested on DB_i.

5.2 Results

We present in Table 1 the reduction methods for which we observed dramatic tree size reduction, often to the detriment of the generalization accuracy. Several remarks can be made from this table:

RNN+C4.5 vs C4.5: The main specificity of the RNN rule is the keeping of misclassified instances. This way to proceed results in a difficulty to build efficient trees because only overlapping and border points are used to identify relevant features. The results of *RNN+C4.5* confirm this drawback: while the average tree size is highly reduced (6.4 vs 12.1 rules on average), that improves the model comprehensibility, the predictive accuracy is not controlled and decreases a lot (73.0 vs 80.3%). Therefore, we can not claim that *RNN+C4.5* is an efficient tree simplification procedure.

RT3+C4.5 vs C4.5: While $RT3$ is particularly suited to reduce storage requirements of a kNN classifier without reducing the kNN predictive accuracy [22, 26], it seems also to be inefficient for simplifying trees. *RT3+C4.5* does not provide actually a good balance between the tree size reduction, *i.e.* the comprehensibility of the model (5.9 rules on average vs 12.1 for *C4.5*), and the control of the predictive accuracy which falls much (from 80.3% to 72.6). A Student paired t-test proves the significant superiority of *C4.5.* This noting is in fact not amazing because $RT3$ is originally optimized for the kNN classifier. The specificities of $RT3$ based on distance calculations (noise-filtering pass removing instances misclassified by their kNN; instance sorting according to the distance from their nearest enemy) make its adaptation to tree simplification difficult.

RC4.5+C4.5 vs C4.5: According to the average result on the 22 datasets, it is amazing to note that $RC4.5$ allows high tree size reduction, but also entailing an accuracy drop (76.7 vs 80.3), that is significant using a Student paired t-test. In fact, we can explain this global result by analyzing datasets in detail. Actually, we note that for 3 datasets (*Car, Pima* and *Xd6*), the decision tree is so reduced that no rule is induced, resulting in a decision rule in favor of the majority class in LS. This explains why accuracy falls much, respectively -20.0, -5.7 and -22.4. By eliminating these 3 datasets, the global performance of RC4.5 is significant on the 19 remaining databases (near 79%).

Dataset	C4.5			RNN+C4.5			RT3+C4.5			RC4.5+C4.5		
	|LS|	# R	Acc.	# pr.	# R	Acc.	# pr.	# R	Acc.	# pr.	# R	Acc.
Australian	552	8.6	85.0	288.4	4.6	81.9	51.1	2.5	83.9	404.2	6.8	85.0
Balance	500	33.0	77.8	140.0	17.5	70.3	44.0	14.5	70.6	405.4	11.2	76.2
Bigpole	400	5.2	63.4	195.3	2.8	57.3	61.4	2.3	57.6	267.2	2.0	62.7
Breast Cancer	558	13.4	94.6	88.0	7.1	85.9	88.0	5.9	85.9	523.4	4.2	93.0
Car	800	19.8	97.6	103.6	10.5	88.2	32.2	8.7	88.6	618.0	1.0	77.6
Dermato	293	8.4	94.6	86.4	4.5	94.6	73.3	7.1	73.6	279.6	8.4	91.3
Ecocardio	104	3.3	54.3	86.2	1.8	48.7	8.9	1.3	51.7	76.4	1.4	58.2
German	800	8.0	70.8	657.6	4.2	70.2	46.4	3.2	65.7	579.6	5	71.3
Glass	170	4.0	75.8	86.1	2.1	67.8	31.7	1.0	57.8	104.8	3.4	76.6
Heart	216	5.8	72.0	102.9	3.0	65.0	32.3	2.5	65.4	147.0	2.6	66.1
Horse	294	11.6	82.6	135.2	6.1	85.5	25.0	2.0	83.8	257.0	5.4	84.4
Ionosphere	148	10.2	86.4	32.0	5.4	68.8	15.2	1.5	90.7	144.0	6.6	85.7
LED	400	16.2	69.0	140.0	8.6	62.3	44.0	7.1	62.6	266.2	12.6	62.9
LED2	400	7.2	88.4	152.0	3.8	74.4	32.0	5.0	74.4	354.6	6.0	88.1
LED+17	400	18.5	81.5	188.0	9.8	81.0	46.7	18.5	53.5	308.8	4.6	79.4
Pima	614	3.4	71.2	321.4	1.8	63.9	32.8	3.4	70.6	345.0	1.0	65.5
Tic-Tac-Toe	480	17.4	80.4	174.4	9.2	64.7	131.2	4.2	66.7	350.8	4.0	72.5
Vehicle	676	9.2	74.8	300.8	4.9	66.1	87.9	23.5	75.7	504.2	5.6	71.0
Waves	400	33.8	75.6	169.3	17.9	65.6	69.3	1.4	69.6	347.4	22.2	74.8
White House	348	5.8	94.4	37.0	3.1	92.0	22.2	2.0	97.9	336.0	2.2	95.2
Wine	141	5.8	91.8	35.3	3.1	76.5	19.7	2.4	71.3	143.6	4.8	88.4
Xd6	479	17.6	84.0	188.0	9.3	75.9	67.1	9.2	79.0	265.6	1.0	61.6
Average	417	12.1	80.3	168.5	6.4	73.0	48.3	5.9	72.6	319.5	6.3	76.7

Table 1: Number of prototypes (# pr.), tree size (number of rules # R) and Predictive accuracy (Acc.) on 22 datasets with 3 data reduction methods; C4.5 is also presented for comparisons

We present in Table 2 the simplification methods which provide a good balance between tree size reduction and accuracy control.
CF+C4.5 vs C4.5: *CF* removes very few instances (8.1% on average) in comparison with the other data reduction techniques. This is not amazing because *CF* is only intended to remove mislabeled instances. On the other hand, this data preprocessing allows high tree size reduction (8.4 vs 12.1) while controlling accuracy (80.2 vs 80.3). These results confirm in fact those presented in [6].

PSB+C4.5 vs C4.5: Our prototype selection method based on boosting also provides interesting results, with smaller trees (9 rules on average vs 12.1), and almost equivalent from a generalization point of view (80.0 vs 80.3).

PSRCG+C4.5 vs C4.5: Incontestably, *PSRCG* seems to be the most efficient algorithm for simplifying trees. Several criteria are actually improved in comparison with the standard *C4.5* algorithm: (i) only 69% of the total number of instances are used for building the tree, that reduces the learning process; (ii) the number of rules is 13.2% smaller than with the standard *C4.5*, that allows a higher model comprehensibility; (iii) the predictive accuracy is improved (80.8% vs 80.3); using a Student paired t-test over accuracies, we found that it is statistically significant with a critical risk near 15%.

Dataset	C4.5			CF+C4.5			PSRCG+C4.5			PSB+C4.5		Random	
	|LS|	# R	Acc.	# pr.	# R	Acc.	# pr.	# R	Acc.	# R	Acc.	# R	Acc.
Australian	552	8.6	85.0	514.4	6.2	85.7	394.6	10.4	83.6	10.4	84.4	16.4	82.0
Balance	500	33.0	77.8	462.8	18.8	79.3	297.0	25.4	79.0	13.4	75.4	23.2	77.0
Bigpole	400	5.2	63.4	348.8	3.0	64.3	354.6	2.6	67.4	7.6	67.6	7.0	64.8
Breast Cancer	558	13.4	94.6	555.2	7.4	93.7	69.6	6.4	93.4	2.0	88.6	4.6	91.6
Car	800	19.8	97.6	787.6	8.6	90.1	337.6	17.8	96.4	10.8	93.0	15.0	94.6
Dermato	293	8.4	94.6	288.4	10.2	91.4	213.4	8.0	94.6	7.0	94.4	6.8	92.2
Ecocardio	104	3.3	54.3	88.2	3.0	68.0	84.3	4.0	57.0	5.0	55.7	5.3	62.7
German	800	8.0	70.8	709.2	7.4	72.6	676.4	6.4	71.6	5.8	68.4	13.8	70.8
Glass	170	4.0	75.8	113.0	3.4	77.3	120.3	3.8	76.5	4.8	75.6	10.6	76.8
Heart	216	5.8	72.0	195.0	4.8	75.5	146.0	8.8	77.5	6.8	78.0	8.8	79.0
Horse	294	11.6	82.6	279.8	5.6	85.7	246.6	8.2	83.6	7.2	85.4	12.4	76.0
Ionosphere	148	10.2	86.4	143.4	7.8	86.2	112.8	9.0	87.0	8.6	88.0	7.8	77.8
LED	400	16.2	69.0	315.8	15.4	69.2	309.8	14.4	70.0	14.4	70.0	15.4	69.6
LED2	400	7.2	88.4	374.8	7.2	87.5	211.4	4.4	87.4	8.2	87.4	6.2	86.0
LED+17	400	18.5	81.5	370.6	10.0	80.7	364.3	17.8	82.5	16.8	81.8	15.0	78.5
Pima	614	3.4	71.2	533.0	4.0	70.6	468.6	3.6	71.2	3.4	71.8	4.2	71.0
Tic-Tac-Toe	480	17.4	80.4	441.4	8.4	72.5	404.6	14.6	78.0	17.0	79.0	22.4	77.0
Vehicle	676	9.2	74.8	587.2	5.6	73.5	566.6	8.4	74.8	11.8	73.8	10.8	72.4
Waves	400	33.8	75.6	383.6	30.2	74.8	297.8	30.6	75.6	12.5	75.6	23.8	70.4
White House	348	5.8	94.4	342.8	3.0	95.7	135.4	4.8	93.6	2.0	95.0	5.2	93.8
Wine	141	5.8	91.8	144.0	4.8	92.8	98.4	4.8	92.8	3.0	87.2	4.6	89.0
Xd6	479	17.6	84.0	446.4	10.0	77.2	404.0	17.8	83.2	20.0	82.4	21.6	79.2
Average	417	12.1	80.3	383.0	8.4	80.20	287.0	10.5	80.8	9.0	80.0	11.9	78.7

Table 2: Number of prototypes (# pr.), tree size (number of rules # R) and predictive accuracy (Acc.) on 22 datasets. Results in the column "Random" are used for the decomposition of PSRCG's effect as proposed by Oates and Jensen

5.3 Analysis of PSRCG

Among the new data reduction methods tested in this paper, $PSRCG$ seems to be the most efficient for simplifying decision trees, and improving generalization accuracy. In this section, we try to explain why this prototype selection method is suited to simplifying trees, and why it deserves future investigations.

5.3.1 Filter Precision

In [6], authors provide a method for evaluating the consensus filter's ability (called *filter precision*) to identify and eliminate mislabeled instances. Consider a learning set artificially corrupted by a given percentage of noise. One defines the 3 following information items:

- the number of instances discarded by the data reduction technique D,

- the number of instances *a priori* corrupted M,
- the number of corrupted instances discarded by the data reduction technique $M \cap D$.

Brodley and Friedl propose the two following estimates $P(E_1)$ and $P(E_2)$, respectively the probability of throwing out good data, and the probability of keeping bad data:

$$P(E_1) = \frac{D-M \cap D}{N-M},$$

where N is the training set size

$$P(E_2) = \frac{M-M \cap D}{M}$$

While the original 22 datasets are not pure and probably already contain noisy data, we decided to calculate $P(E_1)$ and $P(E_2)$ for different artificial noise levels. We corrupted original data successively with 5, 10, 15 and 20% noise. Table 3 reports $P(E_1)$ and $P(E_2)$ averaged over all datasets and all folds.

Noise Level	$P(E_1)$	$P(E_2)$
5%	0.515	0.311
10%	0.494	0.312
15%	0.481	0.327
20%	0.469	0.360

Table 3: PSRCG's filter precision

Contrary to CF, the value of $P(E_1)$ is difficult to interpret for $PSRCG$, which eliminates not only mislabeled instances but also irrelevant and useless examples. Then, D includes other instances than M that explains a relative high value. On the other hand, results for $P(E_2)$ illustrate $PSRCG$'s ability to eliminate mislabeled instances. For small levels of noise, $PSRCG$ finds many mislabeled instances. Actually, with 5% noise, $PSRCG$ find about 70% of them. From a practical point of view, it explains why $PSRCG$ is particularly suited to simplifying decision trees. Even for higher levels of noise, it detects many corrupted examples (about 64% with 25% noise).

5.3.2 Tree Simplification in the Presence of Noise

According to the previous results, $PSRCG$ would not be too sensitive in the presence of noise. We tested this hypothesis on the 22 datasets, corrupting the data by changing the output class of 5% of the learning instances. The experimental method (cross-validation) remains the same. Results are described in Table 4. As expected, $PSRCG$ is still efficient for simplifying decision trees, even in the presence of noise. On average, the number of rules is reduced (11.2 vs 10.4), and the generalization accuracy is slightly improved (75.5 vs 75.7).

5.3.3 PSRCG's Effect on Tree Size Reduction

According to [16], the impact of a given data reduction technique must be assessed through two components: *the part due to the reduction of the training set size, and that which is directly attributable to the selection strategy.* Authors proposed the following criterion to estimate the effect on tree which is due to the reduction in the training set size:

Dataset	C4.5			PSRCG+C4.5		
	|LS|	# R	Acc.	# pr.	# R	Acc.
Australian	552	9.4	83.0	422.6	8.4	84.2
Balance	500	26.8	73.6	302.8	25.8	72.0
Bigpole	400	3.6	61.0	356.6	3.8	62.6
Breast Cancer	558	14.2	90.6	129.2	8.6	88.6
Car	800	16.8	92.2	407.8	18.4	91.6
Dermato	293	3.8	91.8	127.0	4.8	89.6
Ecocardio	104	2.3	59.3	90.7	3.7	59.3
German	800	5.3	66.3	714.3	6.3	68.3
Glass	170	4.2	70.0	109.4	4.6	69.4
Heart	216	5.0	68.3	164.7	6.3	75.3
Horse	294	12.4	80.2	275.0	9.0	79.0
Ionosphere	148	12.6	81.0	74.8	11.2	79.2
LED	400	15.8	69.2	362.8	15.2	68.2
LED2	400	6.3	85.8	266.0	6.0	85.5
LED+17	400	15.0	73.5	369.3	17.5	75.8
Pima	614	3.6	70.0	503.8	3.0	69.2
Tic-Tac-Toe	480	13.8	72.2	415.8	14.0	70.2
Vehicle	676	7.6	71.4	585.2	8.6	71.4
Waves	400	41.4	70.6	317.4	34.8	70.6
White House	348	3.2	89.6	205.8	3.6	89.2
Wine	141	8.3	85.0	106.3	5.8	91.3
Xd6	479	15.0	78.6	432.8	10.6	75.8
Average	417	11.2	76.5	306.4	10.4	76.7

Table 4: Number of prototypes (# pr.), tree size (number of rules # R) and predictive accuracy (Acc.) on 22 noisy datasets

$$Effect_{LS\ size/Tree\ size} = 100 * \frac{(C4.5\ Size - C4.5\ Random\ Size)}{(C4.5\ Size - C4.5\ PSRCG\ Size)}$$

where,

- *C4.5 Size* corresponds to the size (averaged over the 22 datasets) of the tree that $C4.5$ builds on the whole learning set,
- *C4.5 PRSCG Size* is the size of the tree built on the subset *SB1* of instances selected by $PSRCG$,
- *%Kept is* the percentage of learning instances retained by $PSRCG$,
- and *C4.5 Random Size* corresponds to the size of the tree built on the subset *SB2* containing *%Kept* of instances randomly selected,

All these quantities are described in Table 2:

- $C4.5Size = 12.1$

- $C4.5PSRCG\ Size = 10.5$
- *C4.5 Random Size* is deduced from the column *Random* in Table 2, $C4.5Random\ Size = 11.9$

$$Effect_{LS\ size/Tree\ size} = 100 * \frac{(12.1-11.9)}{(12.1-10.5)} = 15.4\%$$

Then, for the 22 datasets listed in Table 2, an average of 15.4% of *PSRCG*'s effect is due to reduction of training set size, and 84.6% is due to *PSRCG*'s strategy for selecting prototypes. Even if, as already mentioned, the quantity $Effect_{LS\ size/Tree\ size}$ is particularly relevant when accuracies are close, we can conclude that substantial reductions in tree size are directly attributable to the *PSRCG* method in comparison with a random reduction. Actually, decision trees are smaller (10.5 vs 11.9) and more accurate (80.8% vs 78.7).

6 Conclusion

In the field of Knowledge Discovery in Databases, the human comprehensibility of models is as important as the accuracy optimization. To address this problem, many methods have been proposed to simplify decision trees and improve their understandability. In this paper, we have studied the contribution of new case selection methods, called prototype selection (PS) algorithms, for simplifying trees. We used PS algorithms differently from what they are originally intended. Actually, we investigated four PS algorithms as data preprocessing in favor of tree simplification, and compared them with the state-of-the-art tree simplification methods.

Many performance measures allow to evaluate the reliability of such methods. Among all the PS methods tested in this article, *PSRCG* has shown on 22 datasets a dramatic efficiency, (i) by reducing the decision tree size, (ii) improving the generalization accuracy, (iii) tolerating the presence of noise. This algorithm optimizes an information measure within a neighborhood graph. It does not depend on a given learning algorithm, is not computationally expensive (only local modifications of the graph are required) and uses a theoretical framework for halting search (for mor details, see [22]). Proceeding this way, we have established, for the first time, a close link between prototype selection and tree simplification.

In this paper, we have also theoretically established the relationship between training set size and tree size, when the decision tree is built with $C4.5$ and its *error-based pruning.* We have shown that the pruning probability converges on 0 when the training set size increases. However, we only dealt with the error-based pruning [19], and future research will have to include new investigations of the other pruning algorithms. Actually, results presented in [16] show that, except *error-based pruning* and *minimum description length-based pruning*[20], the linear relationship between training set size and tree size is not very established.

References

[1] D. Aha, K. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] H. Almuallim and T.G. Dietterich. Learning with many irrelevant features. In *Ninth National Conference on Artificial Intelligence*, pages 547–552, 1991.

[3] J.P. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C.E. Brodley. Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*, 1998.

[4] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J Stone. *Classification And Regression Trees.* Chapman & Hall, 1984.

[5] L.A. Breslow and D.W. Aha. Simplifying decision trees: A survey. *Knowledge Engineering Review*, 1997.

[6] C.E. Brodley and M.A. Friedl. Identifying and eliminating mislabeled training instances. In *Thirteen National Conference on Artificial Intelligence*, 1996.

[7] J. Catlett. Overpruning large decision trees. In *Eleventh International Joint Conference on Artificial Intelligence*, pages 764–769, 1991.

[8] K.J. Cherkauer and J.W. Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In *Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[9] G.W. Gates. The reduced nearest neighbor rule. *IEEE Trans. Inform. Theory*, pages 431–433, 1972.

[10] P.E. Hart. The condensed nearest neighbor rule. *IEEE Trans. Inform. Theory*, pages 515–516, 1968.

[11] G.H. John. Robust decision trees: removing outliers from databases. In *First International Conference on Knowledge Discovery and Data Mining*, pages 174–179, 1995.

[12] G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Eleventh International Conference on Machine Learning*, pages 121–129, 1994.

[13] D. Koller and R. Sahami. Toward optimal feature selection. In *Thirteenth International Conference on Machine Learning*, 1996.

[14] C.J. Merz and P.M. Murphy. Uci repository of machine learning databases. In *Irvine, CA: University of California Irvine, http://www.ics.uci.edu/~mlearn/MLRepository.html*, 1996.

[15] R. Nock and M. Sebban. Advances in adaptive prototype weighting and selection. *International Journal on Artificial Intelligence Tools*, 10 (1-2):to appear, 2001.

[16] T. Oates and D. Jensen. The effects of training set size on decision tree complexity. In *International Conference on Machine Learning*, pages 254–262, 1997.

[17] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[18] J.R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[19] J.R. Quinlan. *C4.5 : Programs for Machine Learning.* Morgan Kaufmann, 1993.

[20] J.R. Quinlan and R. Rivest. Infering decision trees using minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[21] M. Sebban. On feature selection: a new filter model. In *Twelfth International Florida AI Research Society Conference*, pages 230–234, 1999.

[22] M. Sebban and R. Nock. Instance pruning as an information preserving problem. In *Seventeenth International Conference on Machine Learning*, 2000.

[23] D. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Eleventh International Conference on Machine Learning*, pages 293–301, 1994.

[24] Thrun et al. The monk's problem: a performance comparison of different learning algorithms. *Technical report CMU-CS 91-197-Carnegie Mellon University*, 1991.

[25] P.E. Utgoff. An improved algorithm for incremental induction of decision trees. In *Eleventh Internationl Conference on Machine Learning*, pages 318–325, 1994.

[26] D.R. Wilson and T.R. Martinez. Instance pruning techniques. In *Fourteenth International Conference on Machine Learning*, pages 404–411, 1997.

[27] D.R. Wilson and T.R. Martinez. Reduction techniques for examplar-based learning algorithms. In *Machine Learning*, 1998.