APPLIED STOCHASTIC MODELS IN BUSINESS AND INDUSTRY
Appl. Stochastic. Models Bus. Ind. **15**, 277–299 (1999)

THE EFFECTS OF PRUNING METHODS ON THE PREDICTIVE ACCURACY OF INDUCED DECISION TREES

FLORIANA ESPOSITO,† DONATO MALERBA, GIOVANNI SEMERARO AND VALENTINA TAMMA

Dipartimento di Informatica, Università degli Studi di Bari, via Orabona 4, 70126 Bari - Italy

SUMMARY

Several methods have been proposed in the literature for decision tree (post)-pruning. This article presents a unifying framework according to which any pruning method can be defined as a four-tuple (Space, Operators, Evaluation function, Search strategy), and the pruning process can be cast as an optimization problem. Six well-known pruning methods are investigated by means of this framework and their common aspects, strengths and weaknesses are described. Furthermore, a new empirical analysis of the effect of post-pruning on both the predictive accuracy and the size od induced decision trees is reported. The experimental comparison of the pruning methods involves 14 datasets and is based on the cross-validation procedure. The results confirm most of the conclusions drawn in a previous comparison based on the holdout procedure.

KEY WORDS: Induction of decision trees; Decision tree pruning; State space; Cross-validation study

1. INTRODUCTION

Various heuristic methods have been proposed for the construction of a decision tree, among which the most widely known is the top-down approach [1]. In top-down induction of decision trees (TDIDT) it is possible to identify three tasks [2]:

(1) the assignment of each leaf with a class,
(2) the selection of the splits according to a selection measure, and
(3) the decision when to declare a node terminal or to continue splitting it.

The third task is deemed critical for the construction of good decision trees. There are two different ways to cope with it: Either prospectively deciding when to stop the growth of a tree or retrospectively reducing the size of a fully expanded tree by pruning some branches. Methods that control the growth of a decision tree during its construction are called *pre-pruning* methods, while the others are called *post-pruning* methods [3].

* Correspondence to: Prof. Floriana Esposito, Dipartimento di Informatica Università degli Studi, via Orabona, 4, 70126 Bari, Italy.
† E-mail: esposito@di.uniba.it

CCC 1524–1904/99/040277–23$17.50

Received June 1997
Revised May 1999

Many post-pruning (or simply *pruning*) methods have been proposed in the literature, some of which are: *reduced error pruning*, *minimum error pruning*, *pessimistic error pruning*, *critical value pruning*, *cost-complexity pruning*, and *error-based pruning*. A previous comparative study has already pointed out both their similarities and their differences and investigated the real effect of some of these methods on both the predictive accuracy and the size of the induced tree [4, 5]. In that study, optimally pruned trees have been used to evaluate the maximum improvement produced by an ideal pruning algorithm.

The main purpose of this article is that of providing a further comparison of these pruning methods. Their search spaces and search strategies are investigated, in order to analyse their computational complexity, as well as to point out the biases that affect the search strategies.

In the next section, the search space of pruning methods is introduced. It can be represented as a directed acyclic graph whose nodes correspond to simplified versions of a tree $T_{\max}$, while its edges correspond to the application of a pruning/grafting operator. By defining an evaluation function on the set of graph vertices, the problem of tree pruning can be cast as the problem of searching the vertex with the highest value of f. Section 3 is devoted to the presentation of some well-known pruning methods in the unifying framework of search in the state space. Finally, the data sets considered, the experimental procedure based on cross-validation, and the experimental results are reported in Section 4.

2. A UNIFYING FRAMEWORK FOR DESCRIBING PRUNING METHODS

Before discussing the search space explored by pruning methods, some useful notations are introduced.

A (*rooted*) *tree* can be formally defined as a finite set of *nodes*, $\mathcal{N}_T = \{t_0, t_1, \ldots, t_n\}$, and an associated relation $\mathcal{B}_T \subseteq \mathcal{N}_T \times \mathcal{N}_T$ for which the following properties hold:

(1) there exists only one node t_0 in $\mathcal{N}_T$, named *root*, such that $\forall \langle t_i, t_j \rangle \in \mathcal{B}_T : t_j \neq t_0$;
(2) $\forall t_j \in \mathcal{N}_T$, $t_j \neq t_0$, there exists only one node $t_i \in \mathcal{N}_T$ such that $\langle t_i, t_j \rangle \in \mathcal{B}_T$.

The set $\mathcal{N}_T$ can be partitioned into the set $\mathcal{I}_T$ of *internal* nodes and the set $\mathcal{L}_T$ of *leaves*. A particular subset of $\mathcal{I}_T$ is $\mathcal{E}_T$, the set of internal nodes whose children are all in $\mathcal{L}_T$. Following Breiman *et al.*'s notation, we will denote with T_t the *branch* of T containing t and all its descendants (see Figure 1).

Given a set $\mathcal{O}$ of N observations O_i, each of which is described by an $(M + 1)$-dimensional feature vector, $\langle X_1, \ldots, X_M, C \rangle$, it is possible to build tree-structured classifiers, named *decision trees*.

Definition 1 (Decision tree). A decision tree is a particular tree T in which:

- each node t is associated with a subset of $\mathcal{O}$, $\mathcal{O}(t)$;
- the root t_0 is associated with $\mathcal{O}$ itself;
- every edge $\langle t_i, t_j \rangle \in \mathcal{B}_T$ is labelled with a test result, $L_T(\langle t_i, t_j \rangle)$;
- test results for edges out coming from a node t define a partitioning of the feature space, and hence of $\mathcal{O}(t)$.

If p is a path from the root t_0 of a tree T to a leaf t_i of T,

$$p = \langle t_0, t_1 \rangle, \langle t_1, t_2 \rangle, \ldots, \langle t_{i-1}, t_i \rangle$$

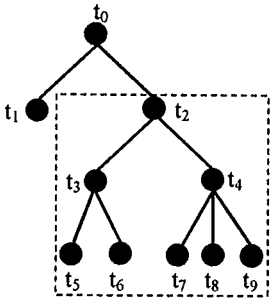

Figure 1. In this tree T, $\mathcal{N}_T = \{t_0, t_1, \ldots, t_9\}$, $\mathcal{B}_T = \{\langle t_0, t_1\rangle, \langle t_0, t_2\rangle, \langle t_2, t_3\rangle, \langle t_2, t_4\rangle, \langle t_3, t_5\rangle, \langle t_3, t_6\rangle, \langle t_4, t_7\rangle, \langle t_4, t_8\rangle, \langle t_4, t_9\rangle\}$, $\mathcal{L}_T = \{t_1, t_5, t_6, t_7, t_8, t_9\}$, $\mathcal{I}_T = \{t_0, t_2, t_3, t_4\}$, $\mathcal{E}_T = \{t_3, t_4\}$. The dashed line encloses the branch T_{t_2}.

then the label associated to p, $L_T(p)$, can be defined as the sequence of labels:

$$L_T(p) = L_T(\langle t_0, t_1\rangle), L_T(\langle t_1, t_2\rangle), \ldots, L_T(\langle t_{i-1}, t_i\rangle)$$

Let $\mathcal{T}$ denote the set of all possible decision trees that can be built from $\mathcal{C}$. It is possible to define two distinct partial-order relations (unless node renaming) on $\mathcal{T}$, namely $\leqslant_{\mathrm{P}}$ and $\leqslant_{\mathrm{G}}$, that satisfies the properties of reflexivity, antisymmetry and transitivity.

Definition 2 (Pruning ordering). Let T and T' be two decision trees in $\mathcal{T}$. Then $T' \leqslant_{\mathrm{P}} T$ iff for each path p' from the root of T' to a leaf in T' there exists a path p from the root of T to a leaf of T such that $L_{T'}(p)$ is a prefix of $L_T(p)$, that is $L_{T'}(p)$ is obtained from $L_T(p)$ by dropping the *last* labels in the sequence.

Definition 3 (Grafting ordering). Let T and T' be two decision trees in $\mathcal{T}$. Then $T' \leqslant_{\mathrm{G}} T$ iff for each path p' from the root of T' to a leaf in T' there exists a path p from the root of T to a leaf of T such that $L_{T'}(p)$ is obtained from $L_T(p)$ by dropping *some* labels in the sequence.

Intuitively, $T' \leqslant_{\mathrm{P}} T$ since T' is obtained from T by *pruning* a branch, while $T'' \leqslant_{\mathrm{G}} T$ when T'' is obtained from T by *pruning* and *grafting* a branch (see Figure 2).

2.1. The search spaces of pruning methods

Given a tree T, it is possible to define two sets of trees, namely:

$$\mathcal{S}_{\mathrm{P}}(T) = \{T' \in \mathcal{T} \mid T' \leqslant_{\mathrm{P}} T\}$$

$$\mathcal{S}_{\mathrm{G}}(T) = \{T' \in \mathcal{T} \mid T' \leqslant_{\mathrm{G}} T\}$$

Obviously, $\mathcal{S}_{\mathrm{P}}(T) \subseteq \mathcal{S}_{\mathrm{G}}(T)$, since $T' \leqslant_{\mathrm{P}} T$ implies $T' \leqslant_{\mathrm{G}} T$. For some subsets of the two partially ordered sets $(\mathcal{S}_{\mathrm{P}}(T), \leqslant_{\mathrm{P}})$ and $(\mathcal{S}_{\mathrm{G}}(T), \leqslant_{\mathrm{G}})$, it is possible to prove some interesting properties. In particular, if $t \in \mathcal{I}_T$, then the set

$$\mathcal{P}_T(t) = \{T' \in \mathcal{S}_{\mathrm{P}}(T) \mid \mathcal{N}_{T'} \subseteq (\mathcal{N}_T - \mathcal{N}_{T_t}) \cup \{t\}\}$$

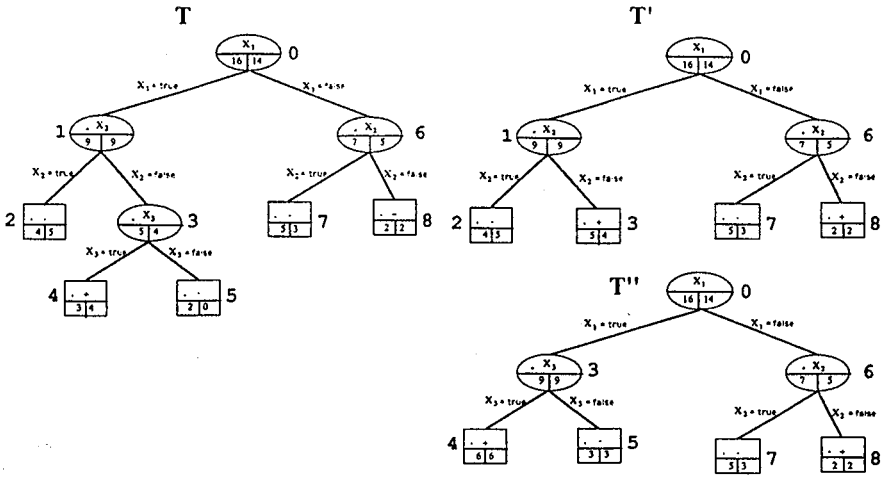

Figure 2. Let us consider the path p' from node 0 to node 3 of T', then $L_{T'}(p') = (X_1 = \text{true}), (X_2 = \text{false})$. This label is obtained by dropping the last node label in $L_T(p) = (X_1 = \text{true}), (X_2 = \text{false}), (X_3 = \text{false})$, where p is the path from node 0 to node 5 in T. By repeating the same check for all paths in T' we can conclude that $T' \leqslant_{\mathrm{P}} T$. Analogously, we have that the label associated to the path p'' from node 0 to node 5 in T'' is: $L_T(p'') = (X_1 = \text{true}), (X_3 = \text{false})$ which is obtained from $L_T(p)$ by dropping the second node label. Repeated checks on all paths in T'' will lead to conclude that $T'' \leqslant_{\mathrm{P}} T$.

has a minimum and a maximum element. The former is the *root tree*, which is made up of the only root node, while the latter is the tree, denoted by $\pi_T(t)$, that have all the nodes of T except the descendants of t. Therefore, we can now define a function π_T:

$$\pi_T : \mathscr{I}_T \to \mathscr{T}$$

that associates each node $t \in \mathscr{I}_T$ with the tree $\pi_T(t)$. This function is called *any-depth branch pruning operator*, since it returns the subtree of T with a pruned branch of any depth.

The *one-depth branch pruning operator*, π'_T, is a restriction of π_T to $\mathscr{E}_T$:

$$\pi'_T : \mathscr{E}_T \to \mathscr{T}$$

thus, the only difference between π'_T and π_T consists in the depth of the trees that they are allowed to prune. With reference to Figure 2, we can say that $\pi_T(3) = \pi'_T(3) = T'$.

The grafting operator can be defined in an analogous way. Let t be an internal node of T, $t \in \mathscr{I}_T$, and t' another internal node of the branch T_t. Then, for the following set:

$$\mathscr{G}_T(t, t') = \{T' \in \mathscr{S}_{\mathrm{G}}(T) \mid \mathscr{N}_{T'} \subseteq (\mathscr{N}_T - \mathscr{N}_{T_t}) \cup \mathscr{N}_{T_{t'}}\}$$

which is a set of subtrees of T having no node of the branch T_t, possibly except those of $T_{t'}$, it can be proved that there exist both a minimum and a maximum. Once again, the former is the root tree, while the latter is the tree, denoted as $\gamma_T(t, t')$, that has all the nodes of T except those in

$\mathcal{N}_{T_t} - \mathcal{N}_{T_{t'}}$. Even in this case, it is sensible to define a function γ_T:

$$\gamma_T : \mathcal{I}_T \times \mathcal{I}_T \to \mathcal{T}$$

that associates each couple of nodes $\langle t, t' \rangle$, $t' \in \mathcal{I}_{T_t}$, with the tree $\gamma_T(t, t')$. This function is called *any-depth branch grafting operator*, since it returns the subtree of T with a branch of any depth grafted onto the place of node t. For instance, by looking at Figure 2, we can write $\gamma_T(1, 3) = T''$. It is worthwhile to observe that, according to the definitions given above, pruning and grafting are two complementary operators, since there is no way of obtaining the tree $\pi_T(t)$ from grafting operators and, vice versa, obtaining $\gamma_T(t, t')$ from pruning operators.

The pruning and grafting operators can be used to define the search space of the pruning methods presented in literature [4]. More precisely, given tree $T \in \mathcal{T}$ with n non-terminal nodes ($n = |\mathcal{I}_T|$), let $\pi_T(\mathcal{I}_T)$ denote the set of the n subtrees of T with a pruned branch. When $|\mathcal{I}_T| = 0$, we set $\pi_T(\mathcal{I}_T) = \emptyset$. The branch pruning operation can be in turn applied to a tree $T' \in \pi_T(\mathcal{I}_T)$, provided that $|\mathcal{I}_T| > 0$. The set of subtrees of T obtained by i subsequent branch pruning operations can be recursively defined as follows:

$$\pi_T^i(\mathcal{I}_T) = \begin{cases} \{T\} & \text{if } i = 0 \\ \pi_T(\mathcal{I}_T) & \text{if } i = 1 \\ \bigcup_{T' \in \pi_T^{i-1}(\mathcal{I}_T)} \pi_{T'}(\mathcal{I}_{T'}) & \text{if } i > 1 \end{cases}$$

It is worthwhile to note that $\pi_T^{n+1}(\mathcal{I}_T) = \emptyset$, since the operation of branch pruning can be applied at most n times to a tree with n internal nodes. Therefore, the set of all possible pruned subtrees of T is given by the union of the sets $\pi_T^i(\mathcal{I}_T)$, $0 \leqslant i \leqslant n$, which coincides with $\mathcal{S}_{\mathrm{P}}(T)$, that is

$$\mathcal{S}_{\mathrm{P}}(T) = \bigcup_{i=0}^{n} \pi_T^i(\mathcal{I}_T)$$

Analogously, an operative definition of the space $\mathcal{S}_{\mathrm{G}}(T)$ can be given. Indeed, given a tree T with n internal nodes, let $\gamma_T(\mathcal{I}_T, \mathcal{I}_T)$ denote a set of subtrees of T with a grafted branch. When $n \leqslant 1$, we set $\gamma_T(\mathcal{I}_T, \mathcal{I}_T) = \emptyset$. By repeatedly applying the branch grafting and pruning operations, it is possible to define the set

$$\gamma_T^i(\mathcal{I}_T, \mathcal{I}_T) = \begin{cases} \{T\} & \text{if } i = 0 \\ \pi_T(\mathcal{I}_T) \cup \gamma_T(\mathcal{I}_T, \mathcal{I}_T) & \text{if } i = 1 \\ \bigcup_{T' \in \gamma_T^{i-1}(\mathcal{I}_T)} (\pi_{T'}(\mathcal{I}_{T'}) \cup \gamma_{T'}(\mathcal{I}_{T'}, \mathcal{I}_{T'})) & \text{if } i > 1 \end{cases}$$

which represents the set of subtrees of T obtained by i subsequent branch pruning and grafting operations. Once again, $\gamma_T^{n+1}(\mathcal{I}_T, \mathcal{I}_T) = \emptyset$, thus the set $\mathcal{S}_{\mathrm{G}}(T)$ of trees $T' \leqslant_{\mathrm{G}} T$ is given by the union of the sets $\gamma_T^i(\mathcal{I}_T, \mathcal{I}_T)$, $0 \leqslant i \leqslant n$, that is

$$\mathcal{S}_{\mathrm{G}}(T) = \bigcup_{i=0}^{n} \gamma_T^i(\mathcal{I}_T, \mathcal{I}_T)$$

The problem of pruning a decision tree can be cast as a search in a *state space*, where *states* are trees in either $\mathscr{S}_{\mathrm{P}}(T)$ or $\mathscr{S}_{\mathrm{G}}(T)$, and branch pruning and grafting are the only *operators* that can be applied in several ways to each tree in the set of states [6]. In particular, we are able to define three distinct state spaces:

- $(\mathscr{S}_{\mathrm{P}}(T), \{\pi_T\})$: *any-depth branch pruning state space*
- $(\mathscr{S}_{\mathrm{P}}(T), \{\pi'_T\})$: *one-depth branch pruning state space*
- $(\mathscr{S}_{\mathrm{G}}(T), \{\pi_T, \gamma_T\})$: *any-depth branch grafting state space*

All pruning methods considered in this article search in one of these spaces, which can be represented by means of directed acyclic graphs (*dag*'s). For instance, the any-depth branch pruning state space can be viewed as a dag whose vertices are just the elements in $\mathscr{S}_{\mathrm{P}}(T)$ and for each couple $(T, T') \in \mathscr{S}_{\mathrm{P}}(T) \times \mathscr{S}_{\mathrm{P}}(T)$ there is an edge from T to T' if and only if $T' \in \pi_T(\mathscr{I}_T)$, that is T' is obtained by pruning only one branch of T of *any depth*. Of some interest is also the space of *one-depth* branch pruning in which there is an edge between every couple of trees $(T, T') \in \mathscr{S}(T) \times \mathscr{S}(T)$ if and only if T' is obtained from T by pruning a branch having a depth equal to one. Indeed, by inverting the direction of the edges, we get a space that coincides with the lattice $(\mathscr{S}_{\mathrm{P}}(T), \leqslant_{\mathrm{P}})$. An example of the lattice of the *one-depth* branch pruning operations is shown in Figure 3a, while the corresponding state space of the *any-depth* branch pruning operations is reported in Figure 3b. In these spaces, each tree T_j^i is uniquely identified by a pair of indices, i and j, where i denotes the number of leaves of the tree while j discriminates among all the trees with

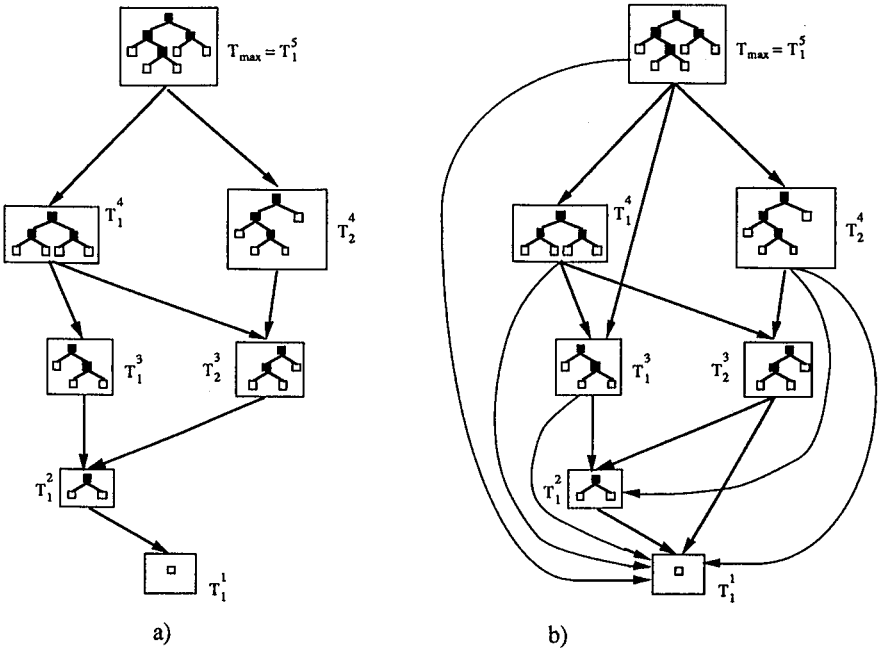

Figure 3. (a) An example of one-depth branch pruning state space. (b) An example of any-depth branch pruning state space.

the same number of leaves. The root tree is denoted by T_1^1, while $T_{\max}$ is denoted as T_1^M, where $M = |\mathscr{L}_{T_{\max}}|$. In $(\mathscr{S}_P(T), \{\pi_T\})$ an edge exists from any $T_j^i \neq T_1^1$ to T_1^1 since T_1^1 can be obtained from T_j^i by applying the pruning operator to the root of T_j^i.

2.2. The evaluation function

In order to give a precise definition of a pruning method the *goal* of the search in the state space has to be defined. For this reason, a function f that estimates the 'goodness' of a tree is introduced. It associates each tree in a generic space $\mathscr{S}(T)$ with a numerical value, namely:

$$f : \mathscr{S}(T) \to \mathscr{R}$$

where $\mathscr{R}$ is the set of real values.

The goal of the search is to find the state in $\mathscr{S}(T)$ with the highest f value, so that pruning can be cast as a problem of function optimization. The formulation of f should depend on both the complexity and the accuracy of a pruned tree.

2.2.1. Evaluating the complexity. In theory, the function f should help to find the smallest decision tree with the highest predictive accuracy. The preference for a small decision tree to a larger one, when both show the same predictive accuracy, is due to a principle popularly known as *Occam's razor*. Indeed, it can be proved that, under very general assumptions, this principle produces hypotheses which with high probability will be as predictive of future observations as predictive of training data [7].

This theoretical conclusion, however, says nothing on how the complexity of a hypothesis should be defined. In the case of hypotheses expressed as decision trees, the complexity is usually measured as follows:

1. number of leaves, $|\mathscr{L}_T|$, or, equivalently, number of regions in which the feature space is partitioned
2. number of internal nodes, $|\mathscr{I}_T|$,
3. average depth of the tree, that is, average number of tests required before taking a decision,
4. topological relevance [8].

More complex measures are obtained by encoding decision trees and the data not explained by the decision trees [9, 10]. Unfortunately, no universal code is known, and, as Wallace and Patrick pointed out, the 'adoption of a particular code for encoding the tree induced from the data is equivalent to accepting a certain prior probability distribution over the set of possible [trees]'. Obviously, an analogous consideration can be applied to the simpler complexity measures listed above.

2.2.2. Evaluating the predictive accuracy. The problem of estimating the error rate of a classifier has received considerable attention in the pattern recognition field [11, 12]. A number of methods have been proposed, which can be classified according to two factors [13]:

1. the way in which examples are used to train the classifier and to test its performance,
2. the *pattern error function* that determines the contribution of an example to the estimate of the probability of misclassification.

One of the simplest ways of using data consists in considering all examples available both for designing the classifier and for evaluating its performances (*resubstitution* method). In the case of tree pruning, this means that the same training set used to build the tree is then exploited to prune it. One of the error rate estimators based on the resubstitution method is the *apparent* (or resubstitution) error rate, computed according to the following pattern error function:

$$e_{\mathrm{A}} = \frac{1}{N} \sum_{i=1}^{N} I\,(d(\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle), C^{(i)})$$

where N is the size of the training set, $\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle$ is the ith observation of the training set, d is a decision rule (in this case a decision tree), I the indicator function:

$$I(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

It is well known that e_{A} is an optimistically biased estimator of the *expected* (or *true*) error rate of a decision rule d,

$$e = E(I(d(\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle), C))$$

This means that $e_{\mathrm{A}} < e$, but it approximates e from below as the size of the training set grows [14].

In order to define unbiased estimators, an alternative way of using data is generally followed. In the *hold-out* method, the data set of examples is partitioned into two subsets, one used for training the classifier and the other one for testing its performance. The pattern error function typically used with the hold-out approach is still the error counting:

$$e_{\mathrm{C}} = \frac{1}{N} \sum_{i=1}^{N} I\,(d(\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle), C^{(i)})$$

where now $\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle$ denotes one of the N test examples. Kittler and Devijver [15] show that the *empirical error count* estimator e_{C} is unbiased and that its variance is given by the following formula:

$$\sigma^2(e_{\mathrm{C}}) = \frac{1}{N}\, e(1 - e)$$

which is actually the variance of a binomial distribution with parameters N and e. This result is important since it shows that, when the true error rate e is small (<0.3) a large number of examples is needed to ensure relatively low variance.

Kittler and Devijver have also studied the properties of the *average conditional* error estimator, e_{R}. In this case, the data set is partitioned into three independent sets, namely the *training* set, the *reference* data set and the *test* set. For each observation $\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle$ in the test set, the k nearest neighbours (k-NN) in the reference data set are used to estimate the conditional

probability of a classification error

$$\hat{P}(d(\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle) \neq C|\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle) = \frac{\tau}{k}$$

where now k is a fixed number of nearest neighbours and τ is the number of these neighbours belonging to classes different from $d(\langle X_1, X_2, \ldots, X_M\rangle)$. The average conditional error estimator is defined as

$$e_{\mathrm{R}} = \frac{1}{N}\sum_{i=1}^{N} \hat{P}(d(\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle) \neq C|\langle X_1^{(i)}, X_2^{(i)}, \ldots, X_M^{(i)}\rangle)$$

It can be proved that, under the assumption of an infinite reference set, e_{R} is unbiased and has a lower variance than the empirical error estimator e_{C}. Nevertheless, in practical situations the reference data set is small, in which case e_{R} may be subjected to a pessimistic bias. Furthermore, it should be observed that not all information in the available data set is actually exploited by this estimator. In particular, information on the class of the testing examples does not affect the computation of e_{R}. In order to avoid this inefficiency, Kittler and Devijver [16] propose a new estimator, denoted as e_{RC}, which is still unbiased under the assumption of an infinite reference set, and has an even smaller variance than e_{R}. Such an improvement is reached by effectively exploiting all information in the data set, but the application of e_{RC} to small samples still remains risky.

Given a dataset of observations, both the empirical and the average conditional error estimators require that part of the data is used for building a tree T, while the rest is exploited to choose the best subtree in a generic space $\mathscr{S}(T)$. In order to avoid the waist of potentially useful information in the test cases, several approaches have been proposed. They are based on the notion of *parametric family* $T(\omega)$ of subtrees of a tree T in a generic state space $\mathscr{S}(T)$. For instance, the parameter ω can be the number of leaves in a subtree of T, or the average depth of a subtree of T. The problem is that of estimating the predictive accuracy of each tree in $T(\omega)$ corresponding to a distinct value of ω.

A first approach consists in *random resampling without replacement* v distinct *validation* sets of the same size from the dataset (*holdout resampling*). Then, from examples not in the ith validation set a new decision tree T_i is generated by means of the same learning algorithm of T. Following the same approach as before, it is possible to define v distinct parametric families, $T_1(\omega), \ldots, T_v(\omega)$, which can help to define the accuracy of the decision trees in $T(\omega)$. More precisely, if $T(\varpi), T_1(\varpi), \ldots, T_v(\varpi)$, are the decision trees with parameter ϖ in the families $T(\omega), T_1(\omega), \ldots, T_v(\omega)$, respectively, then the average of the error rates of the trees $T_1(\varpi), \ldots, T_v(\varpi)$ can be considered an estimate of the error rate of $T(\varpi)$. The strong assumption made in this step is that trees $T_1(\varpi), \ldots, T_v(\varpi)$ have the same error rate as $T(\varpi)$ on the whole feature space. Indeed, while it is sensible to assume that $T, T_1, \ldots, T_v$, have the same error rate when the size of the training sets are not too different, the generalization to the case of subtrees can be hardly justified.

Another different way of building validation sets originates the *v-fold cross-validation* methods. In this case, the observations are randomly divided into v *mutually exclusive cross-validation sets* of approximately equal size. Once again, a tree T_i is built from training examples not in the ith cross-validation set, while its accuracy or, better still, the accuracy of the trees in the parametric

family $T_i(\omega)$, is estimated on the ith cross-validation set. In the extreme case in which v is equal to the number of training examples, the cross-validation estimator is called *leave-one-out* (or *jack-knife*) estimator. Many studies have been performed on cross-validation [17–19] some of which have shown that the leave-one-out provides a nearly unbiased estimate of the true error rate, but often with unacceptably high variability, particularly when the sample size is small [20]. In any case, such results refer to the error estimate of a classifier T given the classifiers $T_1, \ldots, T_v$, built on v cross-validation sets. No study that generalizes such conclusions to parametric families of classifiers, as required in pruning problems, is known.

Finally, in the bootstrap method, training examples for trees T_i are sampled *with replacement* from the data set, thus each single observation can occur more than once in the training set. In order to estimate the accuracy of T, several pattern error functions have been proposed, one of which is simply a linear combination of the average apparent error rate of T_i, $\hat{e}_{\mathrm{A}}$, and the average empirical error rate of T_i, $\hat{e}_{\mathrm{C}}$, that is

$$e_{632} = 0.368\,\hat{e}_{\mathrm{A}} + 0.632\,\hat{e}_{\mathrm{C}}$$

Properties of bootstrap methods for error estimation were first studied by Efron [21], and later work by Jain *et al.* [22] confirmed that they produce more reliable error estimates than the empirical error counter or the cross-validation, not only for Fisher's linear classifiers but also for 1-NN and quadratic classifiers. Similar conclusions were also reached by Crawford [23] for decision trees.

2.3. The search strategy

The way in which the state space is explored can affect both the possibility of finding the best tree and the computational complexity of the method. Actually, any pruning method has to explore a wide space. For a binary balanced tree T of depth d, the number $h(d)$ of trees $T' \leqslant_{\mathrm{P}} T$ is

$$h(d) = \lceil b^{|\mathscr{L}_T|} \rceil = \lceil b^{2^d} \rceil$$

where $b \approx 1.5028368$ and $\lceil x \rceil$ denotes the *ceiling* of a real number x [2].

This means that the size of $\mathscr{S}_{\mathrm{P}}(T)$, when T is a binary balanced tree of depth d, grows double exponentially with d. The size of the search space is even larger for $\mathscr{S}_{\mathrm{G}}(T)$. Thus, the choice of an appropriate search strategy is crucial for the computational complexity of the pruning method.

As a matter of fact, pruning methods presented in this paper adopt only two very simple search strategies: the *first-better search* and the *hill-climbing search.*

In the former strategy, we move from one state T to a state T' just generated if T' is better than T with respect to f. Differently from the hill-climbing, there is no generation of all states directly reachable from T in order to select the best one. Moreover, the first-better strategy does not store all states generated so far as the best-first strategy does. On the contrary, states discarded at a certain moment of the search will no more be considered. Obviously, in this search strategy, the order in which states are generated is of crucial importance. It depends on:

1. the traversal order: *pre-order* or *post-order*,
2. the direction of pruning: *bottom-up* or *top-down.*

In top-down pruning, the root of a branch is considered for pruning before its descendants, while bottom-up pruning starts from leaves and eventually try to prune the root. The bottom-up direction is suitable for post-order traversal, while the top-down direction can be combined well with the pre-order traversal. For instance, with reference to the state space in Figure 3, a top-down method would first explore the state T_1^1, then T_1^2, then T_1^3, and eventually the state T_1^5 by following a pre-order traversal. Conversely, a bottom-up method would first estimate the goodness of the state T_1^5, then of the state T_1^4, then of the state T_2^4, and eventually of the state T_1^1, by following a post-order traversal.

Finally, in some methods the search in the state space is performed in two distinct steps. In the first phase, a tree $T_{\max}$ is pruned up to a fixed degree using an evaluation function f and one of the search strategies listed above. Then, in the second phase, a subset of states traversed in the previous phase is reconsidered for an additional selection on the ground of a new evaluation function f. States not reconsidered are called *transient*. Typically, in the first step of a two-phased search strategy a set of trees of different complexity is collected, while in the second phase the predictive accuracy of collected trees is estimated in order to choose the best tree. In this way, all methods adopting this approach have actually reversed the order in which selection criteria should normally be considered: first predictive accuracy and then, *ceteris paribus*, complexity. The effect is that these methods cannot guarantee the selection of the best, or at least a good, tree with respect to predictive accuracy, which is the most important parameter in classification tasks.

To sum up, a pruning method can be formalized as a 4-tuple:

(Space, Operators, Evaluation function, Search strategy)

Such a scheme allows biases in pruning methods to be clearly identified. In the studies on decision tree induction the following types of biases are identified [24]:

1. *Inductive bias*: methods that search in $\mathscr{S}_{\mathrm{P}}(T)$ have a weaker bias than those searching in $\mathscr{S}_{\mathrm{G}}(T)$.
2. *Preference bias*: for instance, predictive accuracy being equal, a simple decision tree is preferred to a complex one.
3. *Search bias*: methods that do not explore the whole search space may not find the best tree in the space with respect to a given preference bias.
4. *Bias in the estimation of the predictive accuracy*: some estimators of the dual concept of error rate may be biased.

Note that in the first three types, the term bias is intended as a set of factors that influence the hypotheses selection, while in the last one it is intended according to the classical statistical meaning of the term.

The identification of distinct biases of pruning methods provides us with effective means for comparing them analytically. However, the study performed in the paper is not limited to a theoretical investigation of weaknesses and strengths of the various methods, but it will also try to draw some conclusions from some experimental results, as clarified in Section 4.

3. COMPARATIVE STUDY OF PRUNING METHODS

In this section, a comparative study of some well-known post-pruning methods is presented. Each method is cast in the framework of the search in the state space.

3.1. Reduced error pruning (REP)

This method is conceptually the simplest [25] and uses the pruning set in order to evaluate the goodness of a subtree of $T_{\max}$. Search is accomplished in the any-depth branch pruning state space, $(\mathscr{S}_{\mathrm{P}}(T), \{\pi_T\})$, according to the first-better search strategy and a post-order traversal. The evaluation function f is defined as follows:

$$f(T) = -\sum_{t \in \mathscr{L}_T} e(t)$$

where $e(t)$ is the number of errors made by node t during the classification of the examples in the pruning set. The search in the space moves from a state T to a state $T' \in \pi_T(\mathscr{L}_T)$ if the inequality $f(T') \geqslant f(T)$ holds or equivalently if

$$\sum_{t \in \mathscr{L}_{T'}} e(t) \leqslant \sum_{t \in \mathscr{L}_T} e(t)$$

The states to be explored are generated according to the order defined by *bottom-up* methods, hence there is no choice of the best state to be reached, starting from another state. This method finds the *smallest version of the most accurate tree with respect to the pruning set* [4].

Some problems related to this pruning method are the following:

(1) the use of a pruning set distinct from the training set is inadequate when a small number of observations are available, and
(2) the parts of the original tree that correspond to special cases (outliers) not in the pruning set may be lost. Therefore, trees pruned via REP may fail in correctly classifying outliers.

The computational complexity of the method is linear in the number of internal nodes, since each node is visited only once to evaluate the opportunity of pruning it.

3.2. Pessimistic error pruning

This pruning method, proposed by Quinlan [25] as well, is characterized by the fact that it avoids using an independent pruning set. Search is accomplished in the any-depth branch pruning state space, $(\mathscr{S}_{\mathrm{P}}(T), \{\pi_T\})$, according to the first-better search strategy and a pre-order traversal. The evaluation function f is defined as follows:

$$f(T) = -\sum_{t \in \mathscr{L}_T} n'(t)$$

where

$$n'(t) = [e(t) + \tfrac{1}{2}]$$

and $e(t)$ is the number of errors made by node t during the classification of the examples in the training set. Indeed, let T' be the arrival state of an edge outcoming from T such that it is obtained by pruning a node $t \in T$. Then it can be proved that

$$f(T') - f(T) = n'(T_t) - n'(t)$$

where

$$n'(T_t) = -\sum_{s \in \mathscr{L}_T} e(s) + \frac{|\mathscr{L}_{T_t}|}{2}$$

Pruning is accomplished also when the following condition holds:

$$-\mathrm{SE}(n'(T_t)) \leqslant f(T') - f(T)$$

where SE is the standard error. This is equivalent to prune when

$$n'(T_t) + \mathrm{SE}(n'(T_t)) \geqslant n'(t)$$

as stated in Quinlan's original formulation. Therefore, there is no evaluation of the best pruning to perform among the possible ones, and the first pruning operation that turns out to be *good* is performed. It follows that the search strategy adopted is the first-better with a pre-order traversal.

The formulation of the evaluation function can be criticized because of the continuity correction of $e(t)$ in $n'(t)$. Indeed the continuity correction of an error rate has no theoretical justification. Its main effect is that of introducing a tree complexity factor (the constant $\frac{1}{2}$ is the contribute of a leaf to the complexity of the tree), which is improperly compared to an error rate.

It should also be noted that the *top-down* approach to tree pruning used in PEP is not justified when there is no guarantee that all subtrees of a pruned branch T_t have to be pruned. Indeed, it may happen that by pruning a node t other nodes that should not be pruned according to the same criterion are actually discarded. However, this top-down approach gives the pruning method a high run-speed, with a computational complexity being linear in the number of nodes.

3.3. Minimum error pruning (MEP)

Niblett and Bratko [26] proposed a *bottom-up* approach for searching a single tree that minimizes the expected error rate. The proposal has been improved later [27]. For a k-class problem, the expected probability that an observation reaching a node t belongs to the ith class is the following:

$$p_i(t) = \frac{n_i(t) + p_{ai}m}{N(t) + m}$$

where $n_i(t)$ is the number of training examples in t classified into the ith class, p_{ai} is the *a priori* probability of the ith class, m is a parameter of the estimate method, $N(t)$ is the number of training examples reaching t.

When a new observation reaching t is classified, the expected error rate is given by

$$\mathrm{EER}(t) = \min_i \{1 - p_i(t)\}$$

Two error rates are computed for each internal node $t \in \mathscr{I}_T$, namely

1. the *static error*, STE(t), which is the expected error rate of t when pruned, EER(t);
2. the *dynamic* (or *backed-up*) *error*, DYE(t), which is defined as a weighted sum of the expected error rates of the children, where the weights are the probabilities that an observation will reach the corresponding child.

Even for this method, the state space is $(\mathscr{S}_{\mathrm{P}}(T), \{\pi_T\})$, while the search strategy is first-better with a post-order traversal of nodes. The evaluation function of a tree T is the dynamic error of the root of T. It can be proved that such an error equals the weighted sum of the static errors of all the leaves of the tree, where the weights are the proportion of the training examples in the leaves themselves. Formally, we can write

$$f(T) = -\sum_{t \in \mathscr{L}_T} \frac{N(t) \cdot \mathrm{EER}(t)}{N}$$

where N is the total number of training examples.

The search starts with $T_{\max}$ and a new state T_j^i is reached if the following inequality holds:

$$f(T_j^i) \geqslant f(T_{\max}).$$

If T_j^i is obtained by pruning a node t in $T_{\max}$, that is $T_{\max} \in \pi_{T_{\max}}(t)$, then the previous inequality can be equivalently written as

$$\mathrm{STE}(t) \leqslant \mathrm{DYE}(t)$$

which is the condition for pruning a node according to Niblett and Bratko's formulation of the method. Currently, it is still unclear if the minimum error pruning method always finds the maximum in the state space. This aspect will be investigated in the future.

Generally, the higher the m, the more severe the pruning. In fact, when m is infinity, it is $p_i(t) = p_{ai}$ and since p_{ai} is estimated as the percentage of examples of the ith class in the training set, it happens that the tree reduced to a single leaf has the lowest expected error rate. In other words, when m is infinity the path leads to T_1^1. However, this characteristic does not mean that a path corresponding to an $m' > m$ is a continuation of the path corresponding to m. This non-monotonicity property has a severe consequence on the computational complexity: For every different value of m, the search must always start from $T_{\max}$.

In the MEP method, the choice of m is critical. Cestnik and Bratko suggest the intervention of a domain expert who can choose the right value of m according to the level of noise in the data. Alternatively, we have decided to choose the value m using an independent pruning set. More precisely, given a set of possible values for m, we select that returning the smallest tree with the lowest empirical error rate on an independent pruning set. Therefore, this is an example of two-phased pruning method.

3.4. Critical value pruning (CVP)

Mingers [28] has proposed a pruning method that searches in the one-depth branch pruning state space, $(\mathscr{S}_{\mathrm{P}}(T), \{\pi'_T\})$. The evaluation function associated with this reduced space is given by

the sum of the values taken by the selection measure in each internal node of the tree ($f(T_1^1) = 0$ by definition). Therefore, if $\mathrm{GR}(t)$ is the gain ratio at node t, the evaluation function can be defined as

$$f(T) = -\sum_{t \in \mathscr{I}_T} \mathrm{GR}(t)$$

A tree T' will be generated from a tree T if it happens that

$$f(T') = \min_{T'' \in \pi_T'(\mathscr{E}_T)} f(T'')$$

The search goes on according to a *hill-climbing* strategy until the minimum tree T_1^1 is reached. At the end of the search, the number of the explored states will be $|\mathscr{I}_{T_{\max}}| + 1$, denoted as

$$T_{|\mathscr{I}_{T_{\max}}|},\ T_{|\mathscr{I}_{T_{\max}}|-1},\ \ldots,\ T_2,\ T_1,\ T_0$$

This method is two-phased as the previous one. However, in this case not all states traversed are considered in the second phase. A traversed state T_i, $1 < i < |\mathscr{N}_{T_{\max}}|$, is considered to be *transient* if it happens that

$$f(T_j) - f(T_{j-1}) \geqslant f(T_i) - f(T_{i-1}) \quad \forall j, j < i$$

For the second phase, Mingers suggests choosing the best tree among the sequence of the pruned trees by measuring both the significance of the tree as a whole and its predictive ability.

The significance of the tree is estimated by means of the G statistics, which evaluates the degree of interdependence between the leaves of a tree and the classes of the problem: It will be higher for fully expanded trees that correctly classify the whole set of examples. The weakness of this measure is that a test on this statistics is only able to establish whether the predictive ability of a tree is meaningful, but it cannot be used to choose among trees that pass the test.

For what concerns the predictive accuracy of non-transient states, a solution is that of computing the error rate by using an independent pruning set.

However, it should be observed that the sequence detected in the first step of this method might not contain the best tree with respect to the test set, therefore, REP is preferable to CVP, since it guarantees to find the smallest subtree having the lowest error rate with respect to the pruning set.

Finally, the method does not seem sufficiently general to be applied to trees built by using any selection measure. For instance, if the gain-ratio [29] is used as selection measure and the construction of a tree is stopped when all the observations in the training set are correctly classified, it turns out that all nodes in $\mathscr{E}_T$ have a gain-ratio value equal to 1.0, while the others have a lower value. As a consequence, only $T_{\max}$ and T_1^1 will not be considered as transient states, and the choice in the second step will be too restricted.

3.5. *Cost-complexity pruning*

This pruning method is characterized by two phases [2]:

(1) selection of a family of subtrees of $T_{\max}$ according to some heuristics;
(2) choice of the best tree in the family according to an accurate estimate of the actual error rate.

For what concerns the first phase, search is performed in the any-depth branch pruning state space according to a hill-climbing strategy and a post-order traversal. The evaluation function can be defined as follows:

$$f(T) = -\sum_{t \in \tilde{\mathscr{L}}_T} e(t)$$

where $e(t)$ is the number of errors made by node t on the training/growing set. It is possible to move from T to $T' = \pi_T(t)$ if it happens that

$$\frac{f(T) - f(T')}{|\tilde{\mathscr{L}}_T| - |\tilde{\mathscr{L}}_{T'}|} = \min_{T'' \in \pi_T(\mathscr{S}_T)} \frac{f(T) - f(T'')}{|\tilde{\mathscr{L}}_T| - |\tilde{\mathscr{L}}_{T''}|}$$

Indeed, the ratio above can be proved to be equal to

$$\frac{f(T) - f(T')}{|\tilde{\mathscr{L}}_T| - |\tilde{\mathscr{L}}_{T'}|} = -\frac{R(t) - R(T_t)}{|\tilde{\mathscr{L}}_{T_t}| - 1} = \alpha_{T'}$$

which is the *complexity* parameter of the tree T' [2]. For each reached state, the next state that gives the lowest value of the ratio 'apparent error rate increase' on 'number of leaves decrease' is detected. The search goes on until the smallest tree T_1^1 is reached.

The second phase of the method aims at selecting the best among the trees traversed in the first phase. Once again, not all states are considered, and a transient state can be defined as follows: let $T_{\max} = T^m, T^{m-1}, \ldots, T^2, T^1, T^0 = T_1^1$ be the states followed by the search process and let α_T^i be the complexity parameter of a state T^i, then T^i is transient if $\alpha_T^i = \alpha_T^{i-1}$. In other words, if two subsequent trees have the same complexity parameter, only the simpler of the two is considered.

The selection of the best tree is guided by either the empirical error count estimator or the cross-validation error estimator, as described in Section 2.2. In the former case, an independent pruning set is needed, so the tree $T_{\max}$ has to be built on a subset of the training set. In our experiments both approaches have been tested. The methods have been called 0SE and CV-0SE, respectively. As in Breiman *et al.*'s experimental setup, cross-validation is always 10-fold.

For the sake of completeness, another variant proposed by Breiman *et al.* has also been implemented. It is based on the 1SE selection rule, according to which the smallest tree whose error rate estimate e falls within one standard error from the minimum value is chosen. By introducing this variant, other two methods are obtained. They are denoted as 1SE and CV-1SE, respectively.

3.6. Error-based pruning (EBP)

This is the method implemented in C4.5 [29], the learning system used in our experiments. It is considered an improvement of PEP, since it is founded on a far more pessimistic estimate of the expected error rate. As PEP it exploits information in the training set both for building and simplifying trees. However, the novelty is that it searches in the any-depth branch grafting state space $(\mathscr{S}_G(T), \{\pi_T, \gamma_T\})$. The exploration of this space is basically performed according to a first-better strategy. A tree T is visited bottom-up according to a post-order traversal. For each

traversed node t, the following two alternatives are considered:

1. pruning T in t, $\pi_T(t)$, and
2. grafting the *largest* branch $T_{t'}$ of T_t onto the place of t, $\gamma_T(t, t')$ ($T_{t'}$ is the subtree immediately below t that represents most of the examples, not necessarily the largest in size).

If neither of them appears to improve the estimated error rate, than t is left, otherwise the best alternative is chosen and the tree can be either pruned or grafted. Since there is no choice among different nodes the search strategy is still a first-better as for PEP. It is interesting to note that, in this way, not all $\mathscr{S}_G(T)$ is explored, since not all possible grafting operations are considered for reasons of computational economy. The evaluation function used during the search can be defined as follows:

$$f(T) = -\sum_{t \in \mathscr{L}_T} U_{CF}(e(t), N(t)) \cdot N(t)$$

where $U_{CF}(e(t), N(t))$ is the upper limit of a confidence interval, with confidence level CF, that $e(t)$ errors are observed on $N(t)$ training cases, when they are binomially distributed. A new state $T' = \pi_T(t)$ or $T' = \gamma_T(t, t')$ is reached from a state T if it happens that $f(T') \geqslant f(T)$.

Obviously, the search bias remarked above does not guarantee that the best tree is $\mathscr{S}_G(T)$, with respect to $f(T)$, is found. Furthermore, the two assumptions underlying the preference bias adopted for the EBP seem quite strong. Indeed, regarding the training examples covered by a node $t \in T_{max}$ as a statistical sample is hard to accept, since T_{max} is not a generic tree randomly selected from a (possibly infinite) family of decision trees, but it has been built in order to fit the data as well as possible. Even more so, the assumption of a binomial distribution of errors in the sample can be criticized.

4. EMPIRICAL COMPARISON

A previous empirical study on the above pruning methods has already investigated the real effect of some of these methods on both the predictive accuracy and the size of the induced tree [4, 5]. The main conclusions drawn from an experimentation on fourteen datasets available in the UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html) are the following:

(1) MEP, CVP, and EBP tend to underprune, whereas CV-1SE, 1SE, and REP have a propensity for overpruning.
(2) Setting aside some data for pruning is not generally the best strategy.
(3) Pruning does not generally decrease the predictive accuracy of the final trees.
(4) Almost all data sets not prone to pruning have the highest base error, while those data sets with a relatively low base error benefit from any pruning strategy.

The experimental procedure adopted in the previous empirical comparison is the *holdout resampling*, that is the repeated random partitioning of the data into two mutually exclusive subsets called a *training* set and a *test* set. The training and test sets are 70 and 30% of the whole data set, respectively. For each of the 25 runs, the error rate of the pruned tree on the test set is

computed. The estimated error rate of the tree pruned with a specific method is computed by averaging the runs. Then a paired t-test is used in order to compare two methods and detect possibly significant differences in error rate.

The criticism to holdout resampling mainly concerns the assumption of independence of instances in the test set from those in the training set. Thus, the estimates are fine if viewed as comprising a descriptive comparison of performance, but cannot be statistically extrapolated to a wider population. Since in the literature a different experimental procedure has been recommended for model selection [30], namely stratified k-fold cross-validation with moderate k values (10–20), we have decided to repeat our experiments in order to verify whether the conclusions (1)–(4) still hold with cross-validation estimates. In the following we present the new experimental design based on 10-fold cross-validation, new estimates are found and we discuss the conclusions above in the light of the new results.

4.1. Design of the experiment

Each data set D is randomly partitioned into ten *validation sets* (or *folds*) D_i of approximately equal size. Random sampling is *stratified*, so that validation sets contain approximately the same proportions of cases per class as the original datasets. The TDIDT system used, namely C4.5, is trained and tested 10 times; each time $i \in \{1, 2, \ldots, 10\}$, it is trained on $D \backslash D_i$ and tested on D_i. Thus each $D \backslash D_i$ plays the role of *training* set, while D_i that of *test* set. A cross-validation estimate of error rate (size) is obtained by averaging the test error (number of leaves) over the ten folds.

At each run, the training set is split further into two subsets, called *growing* and *pruning* sets. The former contains 70% of cases of the training set, while the latter the remaining 30%. Both the growing and the training set are used to learn decision trees, that are called *grown* tree and *trained* tree, respectively. Grown trees are used by those pruning methods that need an *independent* pruning set in order to prune a decision tree $T_{\max}$, namely REP, MEP, CVP, 0SE and 1SE. Conversely, trained trees are used by those methods that exploit the whole training set for growing and pruning $T_{\max}$, namely PEP, CV-0SE, CV-1SE and EBP. In other words, all methods for building (that is, growing and pruning) a decision tree have access to the same cases, but they can use data in different ways. Some of them prefer setting aside some observations for pruning only, while others prefer using the whole set for the growing phase. In this latter case, the same training set is also used for pruning. Obviously, the error rate of the induced trees is always evaluated on the test sets.

Major properties of the data sets considered in our experiments are summarized in Table I. For each database the following information is reported: The number of cases available in the data set, the number of classes of the training cases, the number of attributes used to describe each example, the number of attributes that are treated as real-valued, the number of non-numerical attributes with more than two values, the presence of null values in the description of an example, the error rate obtained if the most frequent class is always predicted, the expected amount of noise in the database, the uniformity of distribution of training examples per class.

4.2. Experimental results

The first factor to be analysed is the error rate of the pruned trees. In Table II the average error rates concerning different simplification methods are reported together with the average error

Table I. Main characteristics of the data sets used for the experimentation.

Data base	No. cases	No. classes	No. attributes	Real	Multi	Null values	Base error	Noise level	Uniform distrib.
Iris	150	3	4	4	0	no	66.67	low	yes
Glass	214	7	9	9	0	no	64.49	low	no
Led	1000	10	7	0	0	no	90	10%	yes
Hypo	3772	4	29	7	1	yes	7.7	none	no
P.-gene	106	2	57	0	57	no	50	none	yes
Hepatitis	155	2	19	6	0	yes	20.65	none	no
Cleveland	303	2	14	5	5	yes	45.21	low	approx.
Hungary	294	2	14	5	5	yes	36.05	low	no
Switzerland	123	2	14	5	5	yes	6.5	low	no
Long Beach	200	2	14	5	5	yes	25.5	low	no
Heart	920	2	14	5	5	yes	44.67	low	approx.
Blocks	5473	5	10	10	0	no	10.2	low	no
Pima	768	2	8	8	0	no	34.9	?	no
Australian	690	2	14	6	4	yes	44.5	?	approx.

rate of unpruned trained trees. The lowest error rate is reported in bold type, while the highest values are also underlined. We can immediately see that for each data set there is always at least a pruning method that improves the predictive accuracy. Furthermore, in several data sets, such as Hypothyroid, Hungary, Switzerland, Long Beach, Blocks, Pima and Australian, almost all pruning methods reduce the error rate.

In order to verify whether these differences are statistically significant, we used the two-tailed paired t-tests between the pruned decision trees and the trained trees. Table III reports the outcomes of the tests for a significance level equal to 0.10. A '+' in the table means that, on average, the application of the pruning method actually improves the predictive accuracy of the decision tree, while a '−' indicates a significant decrease in predictive accuracy. When the effect of pruning is neither good nor bad, a 0 is reported. From a quick look at Table III we can confirm our previous conclusion that tree pruning does not generally decrease the predictive accuracy. The data sets that do not benefit from pruning are those with the highest base error. It is also confirmed that setting aside some data for pruning is not generally the best strategy. This latter conclusion is at variance with that reported by Mingers [31] in a previous empirical comparison. Finally we observe that the number of statistically significant differences (positive or negative) in this experimentation is lower than that reported in previous work. This is due to the fact that now 90 per cent of data are used for the training set, while in previous experiments the decision tree inducer had access to only 70 per cent of the data.

The results of the two-tailed paired t-tests between the size of trees produced by a pruning method with the size of the corresponding optimal trees [4, 5] are reported in Table IV. Once again, the significance level used in the test is 0.10. Here, 'u' stands for significant underpruning, 'o' for significant overpruning, while '−' means no significant difference. At a glance we can say that MEP, CVP, and EBP tend to underprune, whereas CV-1SE and 1SE have a propensity for overpruning. The tendency of REP to overpruning is less evident because now the growing and pruning sets are wider than those used in the previous experiment.

Table II. Average error rates of pruned trees.

Data base	REP	MEP	CVP	0SE	1SE	PEP	CV-0SE	CV-1SE	EBP	Trained
Iris	5.335	5.335	**4.001**	5.335	5.335	6.669	4.668	**19.999**	4.668	5.334
Glass	29.982	29.915	**20.005**	29.417	33.636	30.998	32.9	**35.715**	31.473	31.473
Led	27.1	27.5	27.5	28	**30**	28.3	27.2	29.2	**26.5**	27.6
Hypo	0.452	0.559	**0.69**	0.505	0.611	**0.452**	0.505	0.452	0.453	0.532
P.-gene	23.544	27.271	**28.362**	23.635	23.544	16.999	23.544	23.544	**14.272**	19.908
Hepatitis	**23.834**	23.167	21.834	22.584	22.584	21.292	21.876	23.167	**18.043**	19.959
Cleveland	31.065	30.387	31.022	32.376	**32.687**	29.688	30.354	32.3	**29.677**	30.011
Hungary	20.412	21.459	**24.908**	21.149	20.447	21.148	**19.069**	19.402	21.459	22.447
Switzerland	8.139	**13.909**	13.076	7.305	**6.536**	**6.536**	**6.536**	**6.536**	**6.536**	13.845
Long Beach	29.5	29	**32.5**	28	26.5	26	25.5	**25.5**	29	32.5
Hearts	**21.631**	22.718	23.045	24.131	**24.783**	22.933	22.39	23.043	21.956	22.392
Blocks	3.325	3.545	**3.8**	3.308	3.381	**3.014**	3.197	3.674	3.215	3.653
Pima	26.826	28.138	**31.39**	26.701	27.09	27.356	**24.614**	26.954	25.66	29.168
Australian	16.52	14.781	**17.825**	**14.364**	14.491	14.636	14.926	14.781	17.245	18.695

Table III. Results of the tests on error rates.

Data base	REP	MEP	CVP	0SE	1SE	PEP	CV 0SE	CV 1SE	EBP	Total +/−
Iris	0	0	0	0	0	0	0	−	0	0/1
Glass	0	0	0	0	0	0	0	0	0	0/0
Led	0	0	0	0	−	0	0	0	0	0/1
Hypo	+	+	0	+	0	0	0	0	0	3/0
P.-gene	0	0	0	0	0	0	0	0	+	1/0
Hepatitis	0	0	0	0	0	0	0	0	0	0/0
Cleveland	0	0	0	0	0	0	0	0	0	0/0
Hungary	0	0	0	0	0	0	0	0	0	0/0
Switzerland	+	0	0	+	+	+	+	+	+	7/0
Long Beach	0	0	−	0	0	+	+	+	0	3/1
Heart	+	0	0	0	0	0	0	0	0	1/0
Blocks	+	+	+	+	+	+	0	0	+	7/0
Pima	+	+	0	+	+	0	+	0	+	6/0
Australian	0	+	0	+	+	+	+	+	0	6/0
Total +/−	5/0	4/0	1/1	5/0	4/1	4/0	4/0	3/1	4/0	

Table IV. Results of the tests on tree size.

Data base	REP	MEP	CVP	0SE	1SE	PEP	CV 0SE	CV 1SE	EBP
Iris	—	—	u	—	—	—	u	—	u
Glass	u	u	u	u	—	u	u	—	u
Led	o	—	u	—	o	—	u	o	u
Hypo	u	u	u	u	—	—	—	—	u
P.-gene	u	u	u	u	—	—	—	o	u
Hepatitis	—	u	u	—	—	—	—	o	u
Cleveland	—	u	u	—	o	u	—	o	u
Hungary	—	—	u	—	o	—	o	o	u
Switzerland	—	u	u	—	—	—	—	—	—
Long Beach	—	u	u	o	o	o	o	o	—
Heart	u	—	u	—	o	o	—	o	u
Blocks	u	u	u	—	o	u	o	o	o
Pima	—	u	u	—	o	u	o	o	u
Australian	—	o	u	o	o	u	o	o	u

5. CONCLUSIONS

Determining the leaves of a decision tree is a critical issue in top-down induction of decision trees. Post-pruning is generally preferred to the pre-pruning approach to the problem. As a matter of fact, many post-pruning methods have been proposed in the literature, all of which can be easily investigated by means of a unifying framework presented in the article. This framework has allowed us to show the simplicity of the adopted search techniques, which, in some cases, might not even guarantee to find the best pruned tree with respect to the evaluation function. Moreover,

we have also pointed out that each evaluation function has its own implicit bias, which amounts to a prior probability distribution over the set of pruned trees.

A new comparison of some pruning methods for decision tree induction is also presented in the article. The new experiment based on cross-validation tries to guarantee the independence assumption, violated in holdout resampling, which is the procedure adopted in a previous study. In fact, the current results confirm the previous conclusions: (1) Pruning methods do not significantly decrease the predictive accuracy of the final trees; (2) those datasets with low base error benefit of almost all pruning strategies; (3) CVP, MEP and EBP tend to underprune while 1SE and CV-1SE tend to overprune.

REFERENCES

1. Safavian SR, Landgrebe D. A survey of the decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics* 1991; **21**:660–674.
2. Breiman L, Friedman J, Olshen R, Stone C. *Classification and Regression Trees*. Wadsworth International: Belmont, CA, 1984.
3. Cestnik B, Kononenko I, Bratko I. ASSISTANT 86: a knowledge-elicitation tool for sophisticated users. In *Progress in Machine Learning*, Bratko I, Lavrac N (eds). Sigma Press: Wilmslow, 1987.
4. Esposito F, Malerba D, Semeraro G. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1997; **19**:476–491.
5. Malerba D, Esposito F, Semeraro G. A further comparison of simplification methods for decision-tree induction. In *Learning from Data: Artificial Intelligence and Statistics V*, Fisher D, Lenz H-J (eds), *LNS-112*. Springer: Berlin, 1996; 365–374.
6. Esposito F, Malerba D, Semeraro G. Decision tree pruning as a search in the state space. In *Machine Learning: ECML-93*, Brazdil P (ed.), *LNAI-667*. Springer: Berlin, 1993; 165–184.
7. Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK. Occam's razor. *Information Processing Letters* 1987; **24**:377–380.
8. Van de Velde W. Incremental induction of topologically minimal trees. In *Proceedings of the Seventh International Conference on Machine Learning*, Porter B, Mooney R (eds). Morgan Kaufmann: Austin, 1990; 66–74.
9. Quinlan JR, Rivest LR. Inferring decision trees using the minimum description trees using the minimum description length principle. *Information and Computation* 1989; **80**:227–248.
10. Wallace CS, Patrick JD. Coding decision trees. *Machine Learning* 1993; **11**:7–22.
11. Toussaint GT. Bibliography on estimation of misclassification. *IEEE Transactions on Information Theory* 1974; **20**:472–479.
12. Hand DJ. Recent advances in error rate estimation. *Pattern Recognition Letters* 1986; **5**:335–346.
13. Raudys SJ, Jain AK. Small sample size effects in statistical pattern recognition: recommendations for practitioners and open problems. *Proceedings of the International Conference on Pattern Recognition*, 1990; 417–423.
14. Gascuel O, Caraux G. Distribution-free performance bounds with the resubstitution error rate. *Pattern Recognition Letters* 1992; **13**:757–764.
15. Kittler J, Devijver PA. Statistical properties of error estimators in performance assessment of recognition systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1982; **4**:215–220.
16. Kittler J, Devijver PA. An efficient estimator of pattern recognition system error probability. *Pattern Recognition* 1981; **13**:245–249.
17. Lachenbruch PA. *Discriminant Analysis*. Hafner Press; New York, 1975.
18. Stone M. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society* B 1974; **36**:111–147.
19. Stone M. Asymptotics for and against cross-validation. *Biometrika* 1977; **64**:29–38.
20. Efron B. Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American Statistical Association* 1983; **78**:316–331.
21. Efron B. Bootstrap methods: another look at the jackknife. *Annals of Statistics* 1979; **7**:1–26.
22. Jain AK, Dubes R, Chen C-C. Bootstrap techniques for error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1987; **9**:628–633.
23. Crawford S. Extensions to the CART algorithm. *International Journal of Man-Machine Studies* 1989; **31**:197–217.
24. Utgoff PE. *Machine Learning of Inductive Bias*. Kluwer Academic: Boston, 1986.

25. Quinlan JR. Simplifying decision trees. *International Journal of Man-Machine Studies* 1987; **27**:221–234 (also appeared in *Knowledge Acquisition for Knowledge-Based Systems*, Gaines BR, Boose JH (eds), Academic Press: New York, 1988).
26. Niblett T, Bratko I. Learning decision rules in noisy domains. *Proceedings of Expert Systems 86*. Cambridge University Press: Cambridge, 1986.
27. Cestnik B, Bratko I. On estimating probabilities in tree pruning. *Proceedings of the EWSL-91*. Springer: Berlin, 1991; 138–150.
28. Mingers J. Expert systems—rule induction with statistical data. *Journal of the Operational Research Society* 1987; **38**:39–47.
29. Quinlan JR. *C4.5: programs for Machine Learning*. Morgan Kaufmann: San Mateo, CA; 1993.
30. Kohavi CR. A study of cross validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995; 1137–1143.
31. Mingers J. An empirical comparison of pruning methods for decision tree induction. *Machine Learning* 1989; **4**:227–243.

