Decision Tree Pruning as a Search in the State Space

Floriana Esposito, Donato Malerba*, Giovanni Semeraro
Dipartimento di Informatica, Università degli Studi di Bari
via G. Amendola 173, 70126 Bari, Italy

Abstract. This paper presents a study of one particular problem of decision tree induction, namely (post-)pruning, with the aim of finding a common framework for the plethora of pruning methods appeared in literature. Given a tree T_{max} to prune, a state space is defined as the set of all subtrees of T_{max} to which only one operator, called any-depth branch pruning operator, can be applied in several ways in order to move from one state to another. By introducing an evaluation function f defined on the set of subtrees, the problem of tree pruning can be cast as an optimization problem, and it is also possible to classify each post-pruning method according to both its search strategy and the kind of information exploited by f. Indeed, while some methods use only the training set in order to evaluate the accuracy of a decision tree, other methods exploit an additional pruning set that allows them to get less biased estimates of the predictive accuracy of a pruned tree. The introduction of the state space shows that very simple search strategies are used by the post-pruning methods considered. Finally, some empirical results allow theoretical observations on strengths and weaknesses of pruning methods to be better understood.

1 Introduction

Decision tree induction has been widely investigated both in the area of machine learning and in pattern recognition. In the vast literature concerning decision trees, at least two seminal works have to be cited: [1, 2]. They come from two schools which worked independently of each other on the same problem, and only at the end of the eighties some papers comparing methods and results from different schools and authors appeared in literature [3, 4, 5, 6]. In this paper we investigate one particular problem concerning decision tree induction, namely the determination of which nodes are leaves, with the aim of finding a common framework for the plethora of methods proposed by the two schools.

In fact, there are two different ways to cope with this problem: either deciding when to stop the growth of a tree or reducing the size of a fully expanded tree, T_{max}, by pruning some branches. Methods that control the growth of a decision tree during its development are called *pre-pruning* methods, while the others are called *post-pruning* methods [7]. The former methods suffer from the problem that the decision of arresting the growth of a branch is always based on local information. Consequently, a test that seems not to add any information on the distribution of the examples is always discarded by any pre-pruning method, even if it would be very useful when subsequently combined with some other tests. This problem is also present in some learning systems that learn Horn clauses and exploit information-based heuristics in order to choose the next literal to add [8]. For this reason, given enough data, it is generally preferred to grow a large tree and then to prune those branches that seem superfluous or even harmful with respect to predictive accuracy [9].

* Currently at the Department of Information and Computer Science, University of California, Irvine, CA 92717.

The variety of post-pruning (or simply pruning) methods proposed in literature does not encourage the comprehension of both the common and the individual aspects. In fact, while some methods proceed from the root towards the leaves of T_{max} when they examine the branches to prune (*top-down approach*), other methods follow the opposite direction, starting the analysis from the leaves and finishing with the root node (*bottom-up approach*). Furthermore, while some methods use only the *training set* in order to evaluate the accuracy of a decision tree, other methods exploit an additional *pruning set*, sometimes improperly called test set, that allows them to get less biased estimates of the predictive accuracy of a pruned tree.

In the next section, we introduce the state space as common framework for all the well-known pruning methods. By introducing an evaluation function defined on the set of states, the problem of tree pruning can be cast as the problem of searching the state with the highest value of *f*. In section 3 this framework is exploited in order to comparatively study different pruning methods so to emphasize their strengths and weaknesses. In this comparison, we will always refer to the same example of decision tree in order to show how the methods really work. Finally, some experimental results together with an explanation of the experimental method are presented in section 4.

2. The search space of pruning methods

Henceforth, we assume that the reader is familiar with some basic notions on decision trees. However, for the sake of clearness, we introduce some notations that we use throughout the paper. In particular, $\mathfrak{N}_T$ denotes the set of *non terminal* nodes of a tree T, while $\mathfrak{F}_T$ denotes the set of *leaves* of T. Moreover, T_t denotes the *branch* of T containing a node t and all its descendants.

Let $\mathfrak{T}$ be the set of all trees. Then the *operation of pruning a branch from a tree* T is a function π_T:

$$\pi_T : \mathfrak{N}_T \rightarrow \mathfrak{T}$$

such that each node $t \in \mathfrak{N}_T$ is associated with the tree $\pi_T(t)$ whose set of nodes is $T \setminus (T_t \setminus \{t\})$, where $\setminus$ denotes the set difference. Such tree is named *subtree of T with a pruned branch.* Note that it does not make sense to prune single node trees.

Given a tree $T \in \mathfrak{T}$ with *n* non-terminal nodes ($n=|\mathfrak{N}_T|$), let $\pi_T(\mathfrak{N}_T)$ denote the set of the *n* subtrees of T with a single pruned branch. When $|\mathfrak{N}_T|=0$, we set $\pi_T(\mathfrak{N}_T)=\emptyset$.

The branch pruning operation can be in turn applied to a tree $T' \in \pi_T(\mathfrak{N}_T)$, provided that $|\mathfrak{N}_{T'}|>0$. Then it is possible to define recursively $\pi^i(\mathfrak{N}_T)$ as follows:

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

that is $\pi^i(\mathfrak{N}_T)$ represents the set of subtrees of T obtained by *i* subsequent branch pruning operations.

It is worthwhile to note that $\pi^{n+1}(\mathfrak{N}_T)=\emptyset$, since the operation of pruning a branch can be applied at most *n* times to a tree with *n* non-terminal nodes. Therefore the set of all possible subtrees of T, $\mathfrak{S}(T)$, is given by:

$$\mathfrak{S}(T) = \bigcup_{i=0}^{n} \pi^i(\mathfrak{R}_T)$$

The problem of post-pruning can be cast as a search in a *state space* [14], where *states* are trees in $\mathfrak{S}(T)$ and branch pruning is the only *operator* that can be applied in several ways to each tree in $\mathfrak{S}(T)$. This space can be represented like a directed acyclic graph whose vertices are just the elements in $\mathfrak{S}(T)$ and for each pair $(T,T') \in \mathfrak{S}(T) \times \mathfrak{S}(T)$ there is an edge from T to T' if and only if $T' \in \pi_T(\mathfrak{R}_T)$, that is T' is obtained by pruning only one branch of T. It should be noted that in this space we move from one state to another by pruning a branch of *any depth*. A matter of interest is also the space of *one depth* branch pruning in which there is an edge between every pair of trees $(T,T') \in \mathfrak{S}(T) \times \mathfrak{S}(T)$ 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 $(\mathfrak{S}(T), \leq)$ in which the order relationship is the tree inclusion.

For the sake of clearness, let us consider the decision tree depicted in Fig. 1. Then its corresponding lattice of the *one depth* branch pruning operations is that reported in Fig. 2 while the corresponding state space of the *any depth* branch pruning operations is shown in Fig. 3. 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 the same number of leaves. Obviously, there exists only one tree with one node that can be obtained from T_{max} by means of the (multiple) application of the branch pruning operator, and such a tree is denoted by T_1^1. Moreover if $|\mathfrak{T}_{T_{max}}| = M$, there exists just one tree having M leaves, namely $T_1^M = T_{max}$. A further observation concerns the fact that in any state space there exists an edge 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.

In order to define the *goal* of the search, a function f that estimates the goodness of a tree has to be introduced. It associates each tree in $\mathfrak{S}(T)$ with a numerical value, namely:

$$f: \mathfrak{S}(T) \rightarrow \mathfrak{R}$$

where $\mathfrak{R}$ is the set of real values. Therefore, the goal of the search is to find the vertex in the graph with the highest value for f, that is to find the *best* subtree of T_{max} with respect to f. In this way, the problem of pruning is cast as a problem of function optimization.

Independently of the adopted optimization method, it is necessary to establish the starting point for the search process. All the pruning methods in literature perform a *forward search*, since they start from T_{max} and go through the graph according to the direction of the edges, nevertheless, they adopt different strategies to generate the sequence of states to explore. We classify the pruning methods as follows:

- *Top-down methods* evaluate first the convenience of pruning the root and then its children, until leaves are considered
- *bottom-up methods*, that start their analysis from the leaves and climb up until the root is considered.

Such a classification does not exactly establish the order in which states are visited, yet. Indeed, T_{max} can have several leaves and it is not clear which leaf will be considered initially and which finally. Therefore, a traversal order for the nodes in a tree must be defined [10].

For instance, with reference to the state space in Fig. 3, a top-down method would first

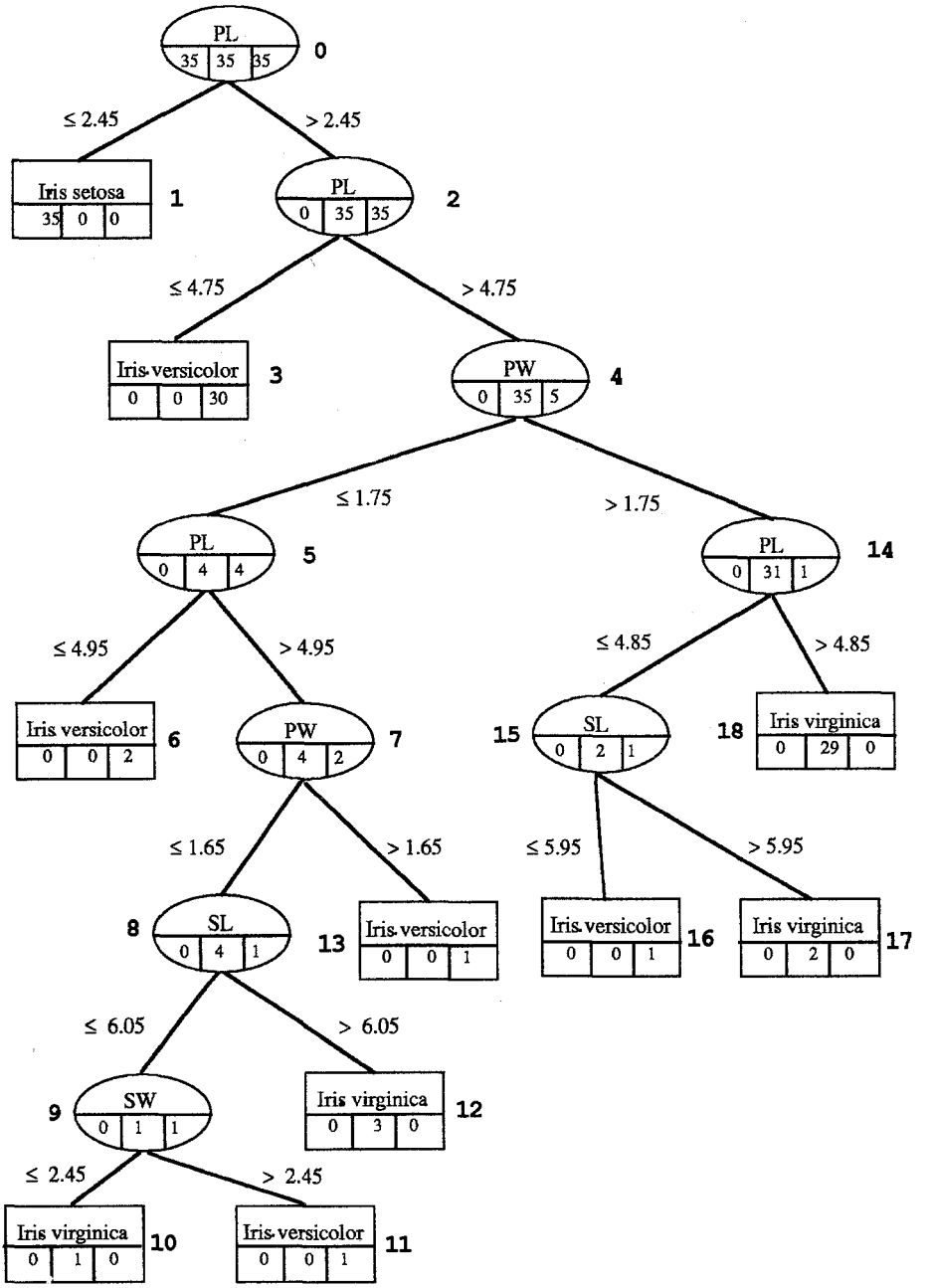

Fig. 1. An example of decision tree induced from a subset of 105 cases of the database "Iris Plants". Observations are classified according to values taken by four distinct continuous attributes: sepal length (SL), sepal width(SW), petal length (PL) and petal width (PW). The nodes of the tree are labelled with numbers assigned by a preorder traversal. Moreover, in each node the number of training instances belonging to each class (*iris setosa*, *iris virginica* and *iris versicolor*) is reported.

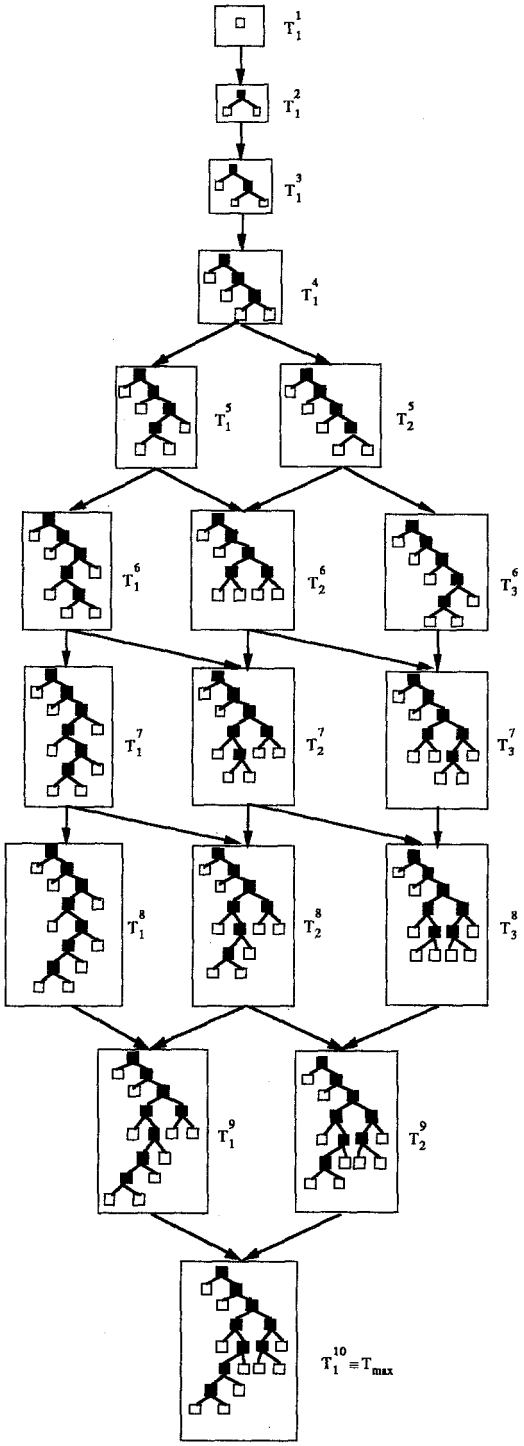

Fig. 2. The lattice of the one depth branch pruning operation for the tree in Fig. 1.

explore the state T_1^1, then T_1^2, then T_1^3, and eventually the state T_1^8 by following a pre-order traversal. Conversely, a bottom-up method would first consider the state T_2^9, then T_3^8, then T_3^7, and eventually T_1^1, by following a post-order traversal.

A further criterion to classify the methods presented in literature is the rationale underlying the definition of the evaluation function *f*. In some methods, *f* is based on statistical information drawn from the *training set* alone, while in other methods information coming from an additional *pruning set* is also exploited. The experimental results reported in [6] show that pruned trees are better according to both predictive accuracy and simplicity when a pruning set is used.

3 Tree pruning as a search in the state space

In this section, a comparative study of five well-known post-pruning methods with respect to the framework of the search in the state space is presented.

3.1 Reduced error pruning

This method, due to Quinlan [3], is conceptually the simplest and uses the pruning set in order to evaluate the goodness of a subtree of T_{max}. It can be easily framed as a search through the state space. Indeed, the evaluation function *f* can be defined as follows:

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

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 $T' \in \pi_T(\tilde{T})$ if the inequality $f(T') \geq f(T)$ holds or equivalently if

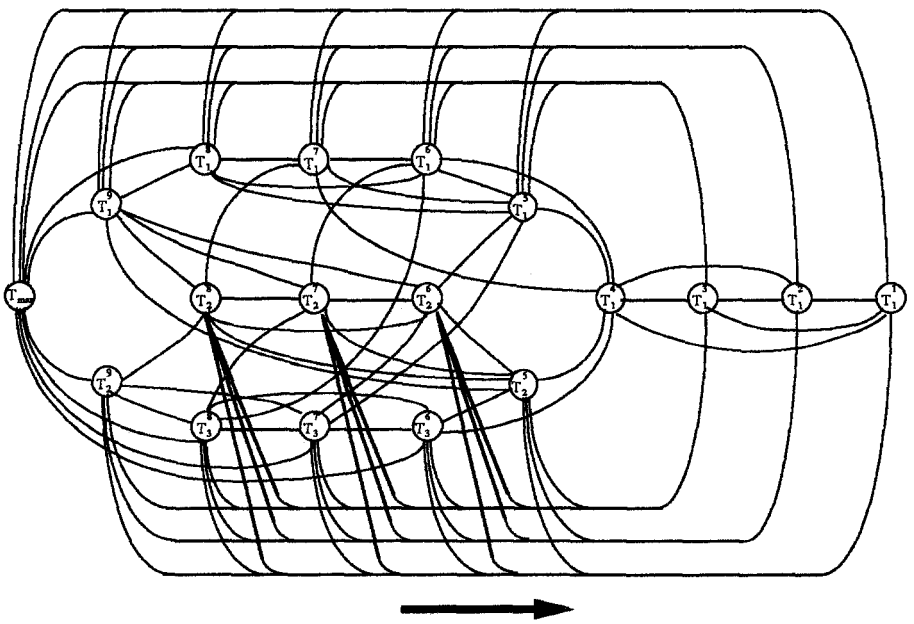

Fig. 3. State space for the tree in Fig. 1. Edges are implicitly directed from left to right.

$$\sum_{t \in \mathbb{T}_{T'}} e(t) < \sum_{t \in \mathbb{T}_T} e(t)$$

Table 1 summarizes the results obtained by applying the reduced error pruning method to the decision tree in Fig. 1 using an independent pruning set of 45 examples. Each non-terminal node is accompanied with the expected error rate for both the case in which the tree undergoes a pruning operation in that node and the case in which no pruning is performed (the corresponding number of classification errors made on the examples of the pruning set is reported in brackets). It is easy to see that, the reduced error pruning method prunes the nodes 9, 8, 7, 15 and 14 since each pruning operation does not increase the error rate of the decision tree. The path followed during the search in the state space is given by:

$$T_{max}, T_2^9, T_3^8, T_3^7, T_3^6, T_1^5.$$

Table 1: Error rates on the pruning set.

# node	error when pruned	error when not pruned
0	66.67% (30)	2.22% (1)
2	33.33% (15)	2.22% (1)
4	4.44% (2)	2.22% (1)
5	4.44% (2)	2.22% (1)
7	2.22% (1)	2.22% (1)
8	2.22% (1)	2.22% (1)
9	2.22% (1)	2.22% (1)
14	2.22% (1)	2.22% (1)
15	2.22% (1)	2.22% (1)

The generation of the states to be explored is made according to the order defined by *bottom-up* methods, therefore there is no choice of the best state to reach, starting from another state. This contrasts with the description of the same method reported in [6]. In fact, Mingers claims that among all the nodes, the one having the largest difference between the number of errors (on the pruning set) when the subtree is kept and the number of errors when the node is pruned must be chosen. This is equivalent to choosing, from all the states of the search space directly connected to the current state, the one with the best value for f. This variant introduces a hill-climbing search strategy instead of the uninformed bottom-up ordered search proposed by Quinlan, thus the guarantee of finding the *smallest version of the most accurate tree with respect to the pruning set* is lost [11].

The problems related to the method of reduced error pruning are basically two:

1) The use of a pruning set distinct from the training set is inadequate when a small number of observations are available
2) the parts of the original tree that correspond to special cases outside the test set may be lost after pruning. Therefore trees pruned via that pruning method may fail in correctly classifying exceptional cases.

At last, it is worthwhile to note that the computational complexity of the method is linear in the number of leaves or exponential in the depth of the tree. Indeed, in the case of a binary balanced tree of depth K, whose best subtree is just that consisting of the root alone, the method of the reduced error pruning will consider all the $2^K - 1$ internal nodes before obtaining the best tree.

3.2 Pessimistic error pruning

This pruning method, proposed by Quinlan [3] as well, is characterized by the fact that it avoids using an independent pruning set. The misclassification rate, estimated by means of the training set alone, results to be optimistic, thus it always happens that pruning

operations based on the same data set produce trees that are larger than it is needed. That reason induced Quinlan to introduce the continuity correction for the binomial distribution that, in his opinion, should provide a more realistic estimate of the classification error rate. Let N(t) be the number of training examples reaching the node $t \in T$ and e(t) the number of them which are misclassified when t is (possibly transformed into) a leaf. Then the number of misclassifications with the continuity correction for the binomial distribution gives:

$$n'(t) = [e(t) + 1/2]$$

for a node t, and:

$$n'(T_t) = \Sigma\, e(s) + |\tilde{T}_t| / 2$$

for a subtree T_t.

It should be observed that, when the development of a tree goes on until all its leaves do not make errors on the training set, e(s)=0 for each $s \in \tilde{T}_t$. In that case n'(T) only represents a measure of the tree complexity that associates each leaf with a cost equal to 1/2. This is no longer true for partially pruned trees or when contradictions (equal observations belonging to distinct classes) occur in the training set.

When a subtree T_t makes fewer errors on the training set than the node t transformed into a leaf, that is $n'(t) \leq n'(T_t)$, the node t can be pruned. Nonetheless, $n'(T_t)$ is still an optimistic estimate, thus Quinlan weakens the condition that rules the pruning of a subtree T_t and requires that:

$$n'(t) \leq n'(T_t) + SE(n'(T_t))$$

where

$$SE(n'(T_t)) = [n'(T_t) \cdot (N(t) - n'(T_t)) / N(t)]^{1/2}$$

is the standard error for the subtree T_t. The algorithm evaluates each node starting from the root of the tree and, if a subtree is chosen for pruning, its internal nodes are not examined. This top-down approach gives the pruning technique a high run speed.

Table 2 reports some data that are relevant for the application of the pessimistic error pruning to the example in Fig. 1. It can be easily seen that the first node to be pruned is node 4, therefore the resultant decision tree has only 3 leaves. In terms of state space, the method moved directly from T_{max} to T_1[3].

The evaluation function associated with each state is the following:

$$f(T) = -\sum_{t \in \tilde{T}} n'(t)$$

In fact, 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 is easy to prove that:

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

that, as told above, is still an optimistic measure for deciding whether to move from a state to another one. This is the reason for which pruning is accomplished also when the following conditions hold:

$$-SE(n'(T_t)) \leq f(T') - f(T) \Leftrightarrow f(T) - f(T') \leq SE(n'(T_t)) \Leftrightarrow n'(T_t) + SE(n'(T_t)) \geq n'(t)$$

Moving from the state T to the state T' occurs when, during the generation of T', the above inequality becomes true. 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. The adopted strategy seems rather poor since the top-down approach to tree

Table 2: Results of the pessimistic error pruning for the tree in Fig. 1.

# node	n'(t)	n'(T_t)	SE(n'(T_t))	$\|\tilde{T}_t\|$
0	70.5	5.0	2.18	10
2	35.5	4.5	2.04	9
4	5.5	4.0	1.89	8
5	5.5	2.5	1.31	5
7	2.5	2.0	1.15	4
8	1.5	1.5	1.15	3
9	1.5	1.0	0.7	2
14	1.5	1.5	1.19	3
15	1.5	1.0	0.81	2

pruning is not justified when there is no guarantee that all subtrees of a pruned branch T_t should be pruned. By looking at data reported in Table 2 it is easy to note that pruning node 4 involves the elimination of the subtree T_5 too, which should not be pruned according to the same criterion. This method has a linear complexity in the number of leaves, and the worst case is that in which the tree has not to be pruned at all.

Lastly, the continuity correction, which has no theoretical justification, simply introduces a sort of tree complexity factor which is improperly compared to an error rate.

3.3 Minimum error pruning

Niblett and Bratko [12] proposed a bottom-up approach seeking a single tree that minimizes the expected error rate on an independent data set. In the following, we will refer to an improved version of the minimum error pruning reported in [13].

For a *k*-class problem, the expected probability of an observation that reaches a node t of belonging to the *i*-th class is the following:

$$p_i(t) = [n_i(t) + p_{ai} \cdot m] / [N(t) + m]$$

where

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

The parameter *m* determines the contribution of the *a priori* probability of the *i*-th class to the estimate of the conditional probability of the *i*-th class in a node t by means of the relative frequency $n_i(t) / N(t)$. For the sake of simplicity, *m* is assumed to be equal for all the classes. Cestnik and Bratko name $p_i(t)$ as *m-probability estimate*. When a new case reaching t is classified, the expected error rate is given by:

$$EER(t) = \min_i \{ 1 - p_i(t) \} = \min_i \{ [N(t) - n_i(t) + (1 - p_{ai}) \cdot m] / [N(t) + m] \}$$

This formula is a generalization of the expected error rate computed by Niblett and Bratko [12]. Indeed, when m=k and $p_{ai}=1/k$, i=1,2,..., *k*, that is the a priori probability distribution is uniform and equal for all classes, we get:

$$EER(t) = \min_i \{ [N(t) - n_i(t) + k - 1] / [N(t) + k] \}$$

which is the formula proposed in the earlier version of the method.

Table 3: Static and dynamic errors for each internal node of the tree in Fig. 1.

	m=1.0		m=3.0		m=10.0	
t	STE(t)	DE(t)	STE(t)	DE(t)	STE(t)	DE(t)
0	66.67%	4.44%	66.67%	9.12%	66.67%	18.59%
2	50.23%	5.73%	50.68%	11.05%	52.08%	20.48%
4	13.82%	8.42%	16.28%	14.79%	23.33%	25.28%
5	51.85%	24.31%	54.55%	39.69%	59.26%	53.41%
7	38.10%	25.00%	44.44%	39.58%	54.17%	52.69%
8	27.78%	23.33%	37.50%	40.00%	51.11%	55.01%
9	55.56%	33.33%	60.00%	50.00%	63.89%	60.61%
14	5.05%	4.44%	8.57%	9.73%	18.25%	20.86%
15	41.67%	25.93%	50.00%	43.33%	58.97%	57.24%

In the minimum error pruning method, the expected error rate for each non-terminal node $t \in \mathfrak{N}_T$ is computed. It is called *static error*, STE(t). Then the expected error rate when t is not pruned, called *dynamic* (or *backed-up*) *error*, DE(t), is computed. It is given by 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. For instance, by referring to the tree in Fig. 1, we have:

$STE(4) = p_5 \cdot EER(5) + p_{14} \cdot EER(14)$

In the original method proposed by Niblett and Bratko, the weights p_i were estimated by the proportion of training examples reaching the *i*-th child. In this case, $p_5=8/40$ while $p_{14}=32/40$. Cestnik and Bratko suggest an *m*-probability estimate with m=2 for p_i, even though the same authors admit that *m* is chosen arbitrarily. However, in this case the *a priori* probability p_{ai} in the *m*-probability estimate refers to the *i*-th node of the tree and not to the *i*-th class. Since it is not clear how it can be computed, in the following we will consider the original proposal, which corresponds to an *m*-probability estimate for p_i with m=0.

In Table 3 values concerning the static error and the dynamic error for each internal node of the tree in Fig. 1 are reported. They have been computed with three distinct values of m: 1.0, 3.0 and 10.0. It is easy to see that when m=1.0 the original tree is not pruned at all, while for m=3.0 only nodes 8 and 14 are pruned and, finally, for m=10.0 nodes 8, 14 and 4 are pruned. Having obtained three distinct trees, the problem of choosing the best one raises. Cestnik and Bratko suggests the intervention of a domain expert who can choose the right value of *m* according to the level of noise in the data or even study the selection of produced trees. Alternatively, when no expert is available, the classification accuracy of the three trees can be evaluated on an independent data set and the smallest tree with the lowest error rate on the pruning set can be selected. In the above example, we get the best results for m=3.0, since the error rate on the pruning set is 2.22%.

The minimum error pruning method can be seen as a way of searching in the state

space. In this case the evaluation function of a tree T can be defined as the dynamic error of the root of T. It can be proven 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:

$$f(T) = -\sum_{t \in \tilde{T}} N(t) \cdot STE(t)/N$$

where

- N is the total number of training examples
- N(t) is the number of examples reaching t.

The search starts with T_{max} and a new state T_j^i is reached if $f(T_j^i) \geq f(T_{max})$. If T_j^i is obtained by pruning a node t in T_{max}, $T_j^i \in \pi_{T_{max}}(t)$, then the previous inequality can be equivalently written as:

$$STE(t) \leq DE(t)$$

which is the condition for pruning a node according to Niblett and Bratko's formulation of the method.

The order of visit of the states is defined *a priori* by the post-order traversal of the bottom-up approach. For instance, when m=3.0 the search path followed by the pruning method in the state space of Fig. 3 is given by: T_{max}, T_3^8, T_1^6. In this example T_1^6 has the highest value of *f*. Nevertheless, it is not clear if the minimum error pruning method always finds the maximum in the state space. It is also worthwhile to observe that, 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 *i*-th 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 gets 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*. In fact the two paths may be completely different, as it is the case of the previous example when m'=10.0. This non-monotonicity property has a severe consequence on computational complexity: for every different value of *m* search must always start from T_{max}.

3.4 Error-Complexity Pruning

The pruning method proposed by Breiman *et al.* [1], the so called *error-complexity pruning*, is characterized by two phases:

1) selection of a family of subtrees of T_{max} according to some heuristics
2) choice of the best tree that belongs to the family by means of an accurate estimate of the actual error rate either on an independent test set or on validation sets.

The method is certainly the most complex among those presented in the paper, nevertheless its description in terms of search in the state space is greatly simplified. Indeed, the evaluation function for $T \in \mathbb{S}(T_{max})$ is given by:

$$f(T) = -\sum_{t \in \tilde{T}} e(t)/N$$

and a state $T' \in \pi_T(\mathbb{R}_T)$ is reached from a state T if it happens that:

$$\alpha_{T'} = [f(T) - f(T')]/[|\tilde{T}| - |\tilde{T}'|] = \min_{T'' \in \pi_T(\mathbb{R}_T)} [f(T) - f(T'')]/[|\tilde{T}| - |\tilde{T}''|]$$

At 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 till the minimum tree T_1^1 is reached. When $T' = \pi_T(t)$ it can be proved that:

$$\alpha_{T'} = - [e(t) - e(T_t)]/[(|\tilde{T}_t| - 1)/N]$$

thus a hill-climbing strategy is adopted to go from state T to state T', which minimizes $\alpha_{T'}$. Table 4 reports the values of $\alpha_{T'}$ for all the $T' \in \pi_{T_{max}}(\tilde{T}_{T_{max}})$, that is the ratio "error rate change" on "number of leaves change" for all the subtrees that can be obtained from T_{max} through one branch pruning operation. It is easy to note that α takes its lowest value in correspondence of nodes 8 and 14, and, since the intersection of T_8 and T_{14} is empty, the tree obtained by pruning both these nodes should be preferred to that obtained by pruning only one of the two nodes. Consequently, the following concept of *transient* state is introduced:

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 ratio "error rate change" on "number of leaves change" for a state T_i, then T_i is transient if $\alpha_{T_i} = \alpha_{T_{i-1}}$. At the end of the *hill climbing* search performed by the *error-complexity pruning* method, only the non-transient states in the path are taken into account to select the best tree.

Going on with the example in Fig. 1, Tables 5, 6 and 7 report the values of α for the states T_1^6, T_1^4, T_1^2 respectively. The complete path followed in the state space by the *error-complexity pruning* method is the following:

$$T_{max}, \mathbf{T_3^8}, T_1^6, \mathbf{T_1^5}, T_1^3, T_1^2, T_1^1$$

where transient states are reported in bold type.

The best subtree selected in the second phase by using the pruning set is T_1^6. This happens because its error rate on this set is 2.22%, that is equal to that of T_{max}, but less than that of T_1^3, T_1^2 and T_1^1. This result does not change if we consider the smallest tree with an error rate within one standard error of the

Table 4: Values of α for the state T_{max}.

t	e(t)	$e(T_t)$	$\lvert\tilde{T}_t\rvert$	α
0	70	0	10	0.074074
2	35	0	9	0.041666
4	5	0	8	0.006803
5	4	0	5	0.009524
7	2	0	4	0.006349
8	1	0	3	0.004762
9	1	0	2	0.009524
14	1	0	3	0.004762
15	1	0	2	0.009524

Table 5: Values of α for the state T_1^6.

t	e(t)	$e(T_t)$	$\lvert\tilde{T}_t\rvert$	α
0	70	2	6	0.129524
2	35	2	5	0.078571
4	5	2	4	0.009524
5	4	1	3	0.014286
7	2	1	2	0.009524

Table 6: Values of α for the state T_1^4.

t	e(t)	$e(T_t)$	$\lvert\tilde{T}_t\rvert$	α
0	70	5	3	0.309524
2	35	5	2	0.285714

Table 7: Values of α for the state T_1^2.

t	e(t)	$e(T_t)$	$\lvert\tilde{T}_t\rvert$	α
0	70	35	2	0.333333

minimum (1SE rule).

By comparing these results with those obtained by the *reduced error pruning* method, it turns out that the tree selected by the error complexity pruning method is not the best one with respect to the pruning set, since it would be possible to prune also node 7 with no error rate increase. By generalizing this result, we can state that this method detects the best subtree in a parametric family of pruned subtrees that might not contain the best tree in an absolute sense or a good one with respect to an evaluation criterion such as the error rate estimated on pruning sets or validation sets.

Finally, the computational complexity of the first phase of the method is the same of the reduced error pruning but the error rate estimate through validation sets requires the building of v auxiliary decision trees.

3.5 Iterative growing and pruning algorithm

Gelfand *et al.*[15] propose a different solution in which all the data set is exploited when growing and pruning a tree and furthermore a search of the optimally pruned tree is performed on the whole set of subtrees. These goals are reached by splitting the data set into two subsets and then by repeatedly growing and pruning a tree on different subsets. More in detail, a tree is grown by using the first subset, then it is pruned by using the second subset. At this point the role of the two subsets is exchanged, so the pruned tree is further grown by using the second subset and pruned with the first one (see Fig. 5).

The authors have also proved a theorem which guarantees the convergence of this iterative process and provides a stopping criterion. In particular they first prove that:

$$T^*_{k-1} \leq T^*_k \quad \forall k=1,2,\ldots$$

where T^*_k is the optimally pruned tree of the k-th iteration (in each iteration a tree is grown by using one subset and then pruned by using the other one). Subsequently they prove that there exists a finite positive number K such that

$$T^*_k = T^*_K \quad \forall k \geq K$$

The proof of this theorem is constructive since it establishes that

$$K = \inf\{\, k \geq 1 : |T^*_{k-1}| = |T^*_k|\}$$

where $|T|$ is the number of nodes in T. Thus the stopping criterion is the following:

$$|T^*_{k-1}| = |T^*_k|$$

When this equality holds for a given k, the tree T^*_k can be accepted as the optimally pruned tree for the entire data set.

The growing phase of this iterative algorithm is accomplished according to the usual recursive partition of the feature space implemented both in CART and in ID3. In their presentation, Gelfand *et al.* employed the GINI index of diversity as selection measure

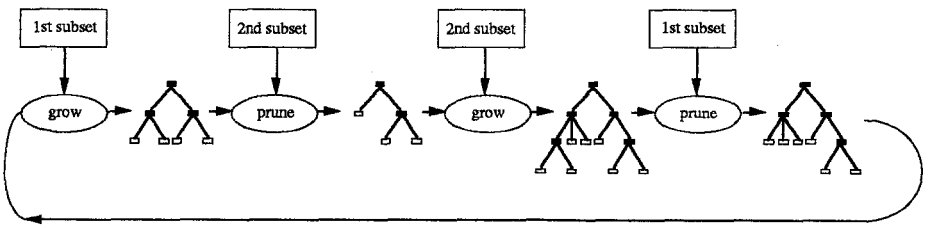

Fig. 5. Schema of the iterative growing and pruning algorithm.

[1] even if this choice does not affect the above theorem. Henceforth, we will use the *gain ratio* [5] to show how this method works.

As to the pruning algorithm proposed by Gelfand *et al.*, it should be noted that it is just the reduced error pruning by Quinlan. Indeed, the estimation of the error rate of each subtree is based on the alternative subset which plays the role of test set. Moreover, the introduction of a cost parameter α as in the error-complexity method, is only a matter of generality in the presentation, since the authors themselves seem to set $\alpha = 0$, thus bringing the evaluation of $R_\alpha(t)$ back to the simple estimation of the error rate on the test set.

A particular attention must be paid to splitting the entire data set into two disjoint sets, since they should be homogeneous as much as possible. In the opposite case, an attribute could show a sufficiently high information gain on the entire data set, but a low information gain in one subset, with the consequence that nodes with tests on that attribute are easily pruned. Such a consideration is also important to point out that the best tree found by this method is not necessarily a subtree of the tree T_{max} grown by using the entire data set.

In order to illustrate this method, the training data set has been split into two subsets: the first 52 examples in the first subset and the remaining 53 in the second one. The tree grown by using the first subset is depicted in Fig. 6a, while the results of the classification of the examples of the second subset are reported in Fig. 6b. According to the reduced error pruning the tree should not undergo any branch pruning operation, thus the second iteration starts just from T_0 and keeps on growing the tree on the second subset until T_1

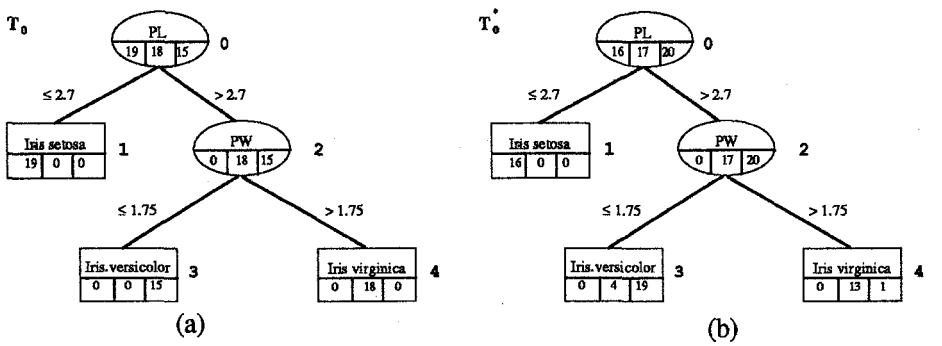

Fig. 6. First iteration: (a) grown tree and (b) classification of cases in the first subset.

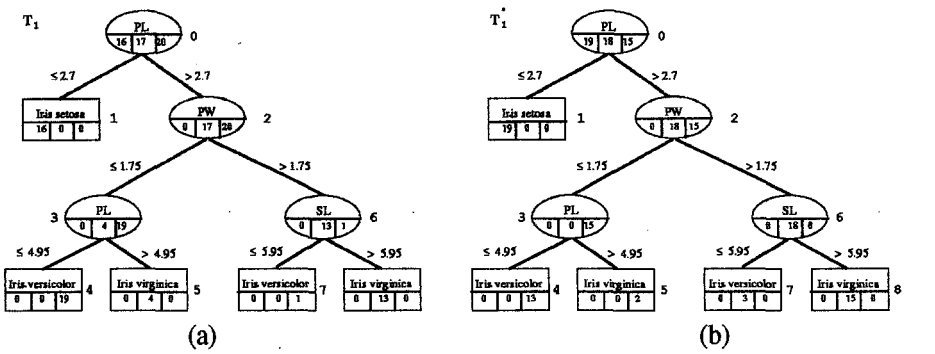

Fig. 7. Second iteration: (a) grown tree and (b) classification of cases in the second subset.

is obtained (see Fig. 7a). By classifying the examples in the second subset, we get the results given in Fig. 7b, therefore the optimally pruned subtree is again T_0. Since trees resulting from two consecutive iterations are exactly alike, the iterative process comes to an end and the final tree is just T_0.

Defining the iterative growing and pruning method in terms of search is more difficult than the other methods. The two phases, growing and pruning, can be individually seen in the framework of search. In particular, the construction of a tree can be cast as search in the (sometimes unlimited) space of all decision trees. However, since we are interested in pruning, we will not investigate this aspect further. As to pruning, we have already presented the reduced error pruning in terms of search in the state space, and those considerations are still valid for this method. In particular, the optimally pruned tree with respect to the pruning set is found. The only difference is that at each iteration the state space may change since the expanded tree may be different from that of the previous iteration. If T_k and T^*_k are the fully expanded tree and the optimally pruned tree of iteration k respectively, then

$$T^*_{k-1} \leq T_k \text{ and } T^*_k \leq T_{k+1} \quad \forall k=1,2,\ldots$$

therefore $\mathbb{S}(T^*_{k-1}) \subseteq \mathbb{S}(T_k)$ and $\mathbb{S}(T^*_k) \subseteq \mathbb{S}(T_{k+1})$. This means that the state space can change at each iteration but all subtrees of T^*_{k-1} are present in the two subsequent state spaces.

A further interesting theorem due to Gelfand *et al.* positively affects the computational complexity of this method. They prove that if a node is a leaf in two consecutive optimally pruned subtrees, T^*_{k-1} and T^*_k, then it will be a leaf in all subsequent optimally pruned subtrees, T^*_i $\forall i \geq k$.

The authors exploit this result during the growing phase, since it is not necessary to split those leaves that satisfy this condition. In our opinion, such a theorem can be extensively exploited in the pruning phase, as well. Indeed, a consequence of this theorem is that it is not necessary to prune all those branches containing at least one leaf appearing in two consecutive optimally pruned subtrees. In other words, the state space for the pruning phase can be reduced by simply marking the ancestors of such leaves.

4 An empirical comparison of pruning methods

The need of making some experiments on some pruning methods arises from the fact that the experimental procedure designed by Mingers [6] to compare several pruning methods is not very fair. Indeed, the author splits each data set into three subsets, a training set (60% of the whole data set), a first test set (20%) and a second test set (20%). The first of these is used by some of the pruning methods, either when a final tree has to be selected from a given family of pruned trees (such as the new version of the minimum error pruning and the error complexity pruning), or in the actual pruning process (reduced error pruning). The second test set is used for measuring the accuracy (or conversely, the error rate) of the resultant trees. Thus, some methods will exploit additional information contained in the first subset to prune, while the others not. We believe that such an unfair experimental procedure may have affected the conclusions drawn out by Mingers in favor of those methods that exploit an independent set for pruning.

Another point concerns the use of the Analysis of Variance (ANOVA) to detect statistically significant differences between pruning methods or between splitting criteria

and interactions between them. Unfortunately, the ANOVA test is based on the assumption that standard deviation is constant for all the experiments and this is not our case because we will compare the algorithms on different data sets. A paired two-tailed t-test for each experiment is the most appropriate way to compare pairs of error averages concerning the same data set.

Similar considerations have also been reported in [16], but they refer to the comparison of splitting rules for decision trees and not to pruning methods. We are also conscious that different pruning methods represents different biases, as already pointed out in [17]. Therefore, we will avoid to draw conclusions on which is the "best" method, but, following the main stream of the analytical comparison between methods followed in the paper, we will try to understand why in some cases a method works better than the others.

In our experimentation, each data set is still randomly divided into three subsets, according to the following criterion:
- *Growing* set (49%)
- *pruning* set (21%)
- *test* set (30%).

The union of the growing and pruning set is called *training* set, and its size is just 70% of the whole data set. The growing set contains the 70% of cases of the training set, while the pruning set the remaining 30%. Both the growing set and the training set are used to learn decision trees, that, for simplicity of naming, we will call *grown* tree and *trained* tree, respectively. The former is used by those methods that need an independent pruning set in order to prune a decision tree, namely reduced error pruning (REP), the new version of the minimum error pruning (MEP), and the error complexity pruning with either the 0SE rule (0SE) or the 1SE rule (1SE). The latter is used by those methods that exploit the training set only, such as pessimistic error pruning (PEP) and iterative growing and pruning algorithm (IGP). The evaluation of the error rate is always made on the test set.

For each data set considered, 20 experiments are made by randomly partitioning the data set into three subsets. Moreover, for each experiment two statistics are recorded:
- The number of leaves (size) of the resultant tree
- the error rate (e.r.) of the tree on the test set.

This is done both for the trees obtained by the different pruning methods (*pruned* trees) and for the grown and trained trees, so that a paired two-tail t-test can be used to evaluate the significance of the error rate difference between pruned trees as well as between each pruned tree and the grown/trained tree. Only the results of those methods that use a prune set will be compared with the results of the grown tree. This is done in order to understand if some results can be partially explained in terms of differences of the tree to prune.

As to the method of minimum error pruning we selected the same values of m used by Niblett and Bratko in their experimentation, that is 0.01, 0.5, 1, 2, 3, 4, 8, 12, 16, 32, 64, 128 and 999. The further partitioning of the training set into two equally sized subsets used by the iterative growing and pruning algorithm is done randomly, as well. However, each subset contains nearly the same proportion of cases per class of the training set, in order to satisfy the conditions presented in [15].

Another difference between our and Mingers' experimentation is that we consider only trees developed according to the gain-ratio (GR) selection measure [5]. Of course, this is a limit for our experimentation, since we cannot detect interesting interactions

between the selection measures and the pruning methods. Future work will need a wider experimentation concerning also this aspect.

4.1 Iris data

This data set contains 150 examples, each of which described by 4 real valued attributes[1]. The examples belong to three different classes corresponding to three different species of iris plants. There are 50 examples per class, thus in each experiment the data set was partitioned as follows: 70 cases in the growing set, 30 cases in the pruning set, and 50 cases in the test set.

The mean and the standard deviation for each statistics concerning the various methods are reported in Table 8, while Table 9 shows the result of the comparison between the error rate of the pruned trees and that of the grown and trained trees (each entry contains the t value and the corresponding significance level). It is worthwhile to note that only the PEP method shows a statistically significant improvement with respect to the trained tree (the probability of error in rejecting the hypothesis that the two mean are equal is less than 0.005). For the other methods, tree pruning seems not to be really effective. On the contrary, if we compare their results with those obtained from the grown tree, we note that pruning tends to slightly increase the error rate.

By pairwise comparing the average error rates obtained by the different methods we can conclude that there is no statistically significant difference between the methods (see Table 10). This means that the best average error rate obtained by PEP is not much better than that of the other average error rates.

Table 8: Mean and standard error for both size and error rate of the resultant trees (iris data).

	REP		MEP		0SE		1SE		grown		PEP		IGP		trained	
	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.
size	3.5	1.0	3.7	1.3018	3.7	1.3018	3.5	1.0513	6.05	1.5381	3.85	0.8127	3.45	0.8256	7.4	2.0365
er.	5.9	2.7891	5.9	2.7891	5.9	2.7891	5.8	2.8946	5.5	2.5026	5.0	3.6992	5.3	3.3888	6	3.4944

Table 9: t value and significance value for the paired two-tailed t-test (iris data)

	REP	MEP	0SE	1SE	PEP	IGP
grown	.809 (.43)	.847 (.41)	.847 (.41)	.616 (.55)	-	-
trained	-.165 (.87)	-.175 (.86)	-.175 (.86)	-.357 (.73)	-3.25 (.004)	-1.23 (.232)

Table 10: results of the paired two-tailed t-tests between methods (iris-data).

	MEP	0SE	1SE	PEP	IGP
REP	.0 (1.0)	.0 (1.0)	.438 (.67)	1.48 (.154)	.9 (.38)
MEP	-	.0 (1.0)	1.0 (.33)	1.53 (.14)	.946 (.36)
0SE		-	1.0 (.33)	1.528 (.14)	.946 (.36)
1SE			-	1.361 (.19)	.793 (.44)
PEP				-	-.459 (.65)

1. Real valued attributes are treated as described in [2].

Finally, there are at least other three points that are worthwhile to note. Firstly, the PEP has the highest average in the number of leaves of the pruned tree. It is possible that its positive bias when pruning is well suited for very regular data like this. Secondly, the IGP always finds the best tree in two steps, that is it needs to develop and prune a tree only twice. The IGP method globally shows good results and is generally able to improve the predictive accuracy of the classifier, but in some experiments it produced the highest error rate. This partly explains its highest standard deviation. In other words, the IGP method seems quite instable.

4.2 Glass data

This data set contains 214 examples, each of which described by 9 real valued attributes plus an identification number that we did not consider. There are 7 classes but no example is provided for the fourth class. In each experiment the data set was partitioned as follows: 105 cases for the growing set, 45 cases for the pruning set and 64 cases for the test set.

Table 11 shows the mean size and the average error rate of each method together with the corresponding standard deviations. By looking at Table 12 it can be seen that there is a statistically significant difference ($p<0.01$) between the error rates given by MEP, 0SE and 1SE, and the error rate of the trained tree. Unfortunately, this difference is positive, meaning that these methods decreased the predictive accuracy of the trained decision tree. A similar result can also be observed for IGP at a .1 significance level. This time the PEP did not actually improve the error rate, as well as REP.

The high error rate of the trained tree show how difficult discovering regularities in this data set is.Perhaps the underlying model is quite complex while data is weak, thus the bias of some pruning methods may cause the very opposite problem of underfitting [17].

5 Conclusions

Determining the leaves of a decision tree is a critical problem of decision tree induction. It can be faced either by (post-)pruning a fully expanded tree or by stopping the growth of the tree itself (pre-pruning). The latter method is generally preferred to the former, since selection measures proposed for decision tree induction are not polythetic, that is the discriminant power of logical combinations of attribute-value pairs is not taken into account. Thus, pre-pruning may stop the growth of a tree because all attributes seem individually meaningless even if some combinations of them may be highly discriminant.

Table 11: Mean and standard error for both size and error rate of the resultant trees (glass data).

	REP		MEP		0SE		1SE		grown		PEP		IGP		trained	
	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.
size	17.15	4.977	23.1	5.17	18.05	8.036	12.25	6.889	29.85	6.738	25.05	3.103	15.35	4.511	39.6	3.202
er.	37.301	6.356	38.752	4.138	39.454	5.142	41.407	6.541	37.265	6.614	34.219	5.409	37.266	6.215	34.295	6.301

Table 12: t value and significance value for the paired two-tailed t-test (glass data)

	REP	MEP	0SE	1SE	PEP	IGP
grown	.028 (.978)	1.171 (.256)	1.304 (.208)	2.11 (.0485)	-	-
trained	1.536 (.141)	2.89 (.0094)	2.974 (.008)	3.44 (.0027)	-0.087 (.93)	1.73 (.1002)

In this paper, a comparative study of six well-known post-pruning methods is presented, together with a unifying view of the tree pruning problem in terms of search in a state space. Given a tree T_{max} to prune, the corresponding state space is defined as the set of all subtrees of T_{max}, $\mathbb{S}(T_{max})$, and the branch pruning operation is the only operator defined on such a set. Each state of $\mathbb{S}(T_{max})$ is associated with the value taken by an evaluation function, f, that assesses the goodness of the tree. Therefore, the problem of pruning may be cast as the problem of visiting different states in $\mathbb{S}(T_{max})$ with the aim of maximizing f.

In the methods considered in the paper, states are always visited according to a pre-established order, depending on the kind of approach: bottom-up or top-down. While in the *reduced error pruning*, *pessimistic error pruning*, and *minimum error pruning*, a branch is pruned as soon as a better state is detected, in the *error complexity pruning* method, the best state among all the directly reachable states is selected according to an hill-climbing search strategy. This last method, together with the *minimum error pruning*, select a parametric family of subtrees of T_{max}, that will undergo a further evaluation process with the aim of detecting the best tree in the family. However, this two-phased selection, thought in order to reduce the computational complexity, has a main drawback: it does not guarantee that the best tree (or a good one) with respect to the predictive accuracy is in the family. A more in-depth analysis of these methods is provided in [11].

Another criterion to discriminate among post-pruning methods is the kind of information exploited by the evaluation function f. Indeed, in the pessimistic error pruning, only information coming from the training set is used, whereas all the other methods need an additional independent test set. A different consideration should be made for the minimum error pruning, since its original formulation used only information coming from the training set and in the newer version the use of an independent pruning set is only hypothesized. When predictive accuracy is estimated by means of cross-validation, the error-complexity pruning does not need a pruning set, as well.

In the paper, some preliminary results concerning a redesigned experimentation on pruning methods have been presented. The first data set contains strong data relative to the simple underlying model. Indeed, the average error rate of the grown tree was better than that of the trained tree. However, all the methods exploiting an independent data set tried to prune further, generally increasing the error rate of the resultant tree. The method that gave a statistically significant improvement on the predictive accuracy is the pessimistic error pruning. The same method, did not improve the classification accuracy of the trained tree for the glass data. In this case data were weak relative to the complex underlying model. Such a result was the best with respect to other pruning methods that even decreased the predictive accuracy.

Future work should consider both a more extensive experimentation in which more databases are considered. Moreover, since the iterative growing and pruning algorithm is affected by the random splitting of the training set, it would be better to evaluate the performance of the method with respect to several random partitioning of the training set, in order to reduce its variance in the corresponding results.

Finally, future research on decision tree pruning should address the problem of defining a less biased evaluation function f to use in a more informed search strategy, since the introduction of the state space has shown that very simple strategies are used by the well-known post-pruning methods.

Acknowledgments

Thanks to M. Pazzani, L. Saitta and A. Giordana for their helpful comments on earlier drafts of the paper.

References

1. L. Breiman, J. Friedman, R. Olshen, C. Stone: Classification and regression trees. Belmont, CA: Wadsworth International 1984
2. J. R. Quinlan: Induction of decision trees. Machine Learning 1, 81-106 (1986)
3. J. R. Quinlan: Simplifying decision trees. International Journal of Man-Machine Studies 27, 221-234 (1987) (also appeared in: B. R. Gaines, J. H. Boose (eds.): Knowledge Acquisition for Knowledge-Based Systems. Academic Press 1988)
4. M. Gams, N. Lavrac: Review of five empirical learning systems within a proposed schemata. In: I. Bratko, N. Lavrac (eds.): Progress in Machine Learning. Wilmslow: Sigma Press 1987
5. J. Mingers: An empirical comparison of selection measures for decision-tree induction. Machine Learning 3, 319 - 342 (1989)
6. J. Mingers: An empirical comparison of pruning methods for decision tree induction. Machine Learning 4, 227 - 243 (1989)
7. B. Cestnik, I. Kononenko, I. Bratko: ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In: I. Bratko, N. Lavrac (eds.): Progress in Machine Learning. Wilmslow: Sigma Press 1987
8. J. R. Quinlan: Determinate literals in inductive logic programming. Proceedings of the IJCAI 91. San Mateo, CA: Morgan Kaufmann 1991, pp. 746-750
9. T. Niblett: Constructing decision trees in noisy domains. In: I. Bratko, N. Lavrac (eds.): Progress in Machine Learning. Wilmslow: Sigma Press 1987
10. A. V. Aho, J. E. Hopcroft, J. D. Ullman: The design and analysis of computer algorithms. Reading, MA: Addison Wesley 1974
11. F. Esposito, D. Malerba, G. Semeraro: Pruning methods in decision tree induction: a unifying view. Technical report (1992)
12. T. Niblett, I. Bratko: Learning decision rules in noisy domains. Proceedings of Expert Systems 86. Cambridge: University Press 1986
13. B. Cestnik, I. Bratko: On estimating probabilities in tree pruning. Proceedings of the EWSL -91. Berlin: Springer-Verlag 1991, pp. 138-150
14. A. Barr, E. Feigenbaum: The handbook of artificial intelligence, (Vol. 1). Reading, MA: Addison Wesley 1981
15. S. B. Gelfand, C. S. Ravishankar, E. J. Delp: An iterative growing and pruning algorithm for classification tree design. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-13, 2, 163-174 (1991)
16. W. Buntine, T. Niblett: A further comparison of splitting rules for decision-tree induction. Machine Learning 8, 75-85 (1992)
17. C. Schaffer: Deconstructing the digit recognition problem. In: Machine Learning: Proceedings of the Ninth International Workshop (ML92). San Mateo, CA: Morgan Kaufmann (1992), pp. 394-399