The J. of MIS
Research
Vol. 3, No. 1, Jun 1993

의사결정나무의 현실적인 상황에서의 팩(PAC) 추론 방법

김 현 수

PAC-Learning a Decision Tree with Pruning

Empirical studies have shown that the performance of decision tree induction usually improves when the trees are pruned. Whether these results hold in general and to what extent pruning improves the accuracy of a concept have not been investigated theoretically. This paper provides a theoretical study of pruning. We focus on a particular type of pruning and determine a bound on the error due to pruning. This is combined with PAC (Probably Approximately Correct) Learning theory to determine a sample size sufficient to guarantee a probabilistic bound on the concept error. We also discuss additional pruning rules and give an analysis for the pruning error.

I. Introduction

The construction of decision trees is an important type of inductive learning [Quinlan, 1986b; Mingers, 1989a; Utgoff, 1989]. This approach may be used directly for predictive or descriptive purposes [Braun and Chandler, 1987; Carter and Catlett, 1987; Messier and Hansen, 1988] or may be applied to knowledge acquisition for expert systems [Quinlan, 1979; Michalski and Chilausky, 1980; Quinlan, 1987b].

Researchers have found that a large decision tree constructed from a training set usually does not retain its accuracy over the whole instance space [Breiman et al., 1984; Quinlan, 1986a; Spangler et al., 1989]. Recently a number of papers investigated pruning large decision trees built from training examples [Quinlan, 1987a; Fisher and Schlimmer, 1988; Mingers, 1989b]. Many of the branches of the constructed large decision trees will reflect chance occurrences in the particular data rather than representing true underlying relationships. Often, these are very unlikely in further examples. Since these less reliable branches can be removed by pruning, the pruned decision tree often gives better classification over the the whole instance space even though it may have a higher error over the training set.

While these papers have reported excellent empirical results of pruning in terms of improving the accuracy of the learned concepts, those results depend heavily on the specific training set and the domains over which they apply. Whether these results hold in general and to what extent pruning improves a concept are unknown.

In this paper we focus on a particular type of pruning and determine its theoretical effect. This is combined with recent results in learning theory to produce a pruning method that will yield a concept that is probably approximately correct (PAC). Decision trees having PAC-property may be used directly

for business decision making or may be applied to knowledge acquisition for expert systems with more rigorous estimates on concept accuracy.

In Section II we review PAC learning. In Section III we present notation and give background material for constructing a decision tree with minimum rank. A pruning algorithm is motivated and presented in Section IV. In Section V we determine a bound on the error due to pruning. In Section VI we give conditions for PAC identification with pruning. In Section VII we discuss additional pruning rules. Finally, in Section VIII we summarize our results and suggest areas for future research.

II.PAC Learning of BinaryDecision Trees

The PAC learning paradigm was introduced by Valiant in 1984. This model requires that a polynomial bounded algorithm identify a concept from a random sample, whose size is polynomially bounded, such that a learned concept has a high probability of being close to the true concept. Angluin and Laird [1988] coined the terminology PAC learning. A more precise definition follows.

Let X be the instance space of interest. The target concept f maps X into {0,1}. Similarly, for any other concept h, we have

h: X → {0,1}.

The error, d(h,f), of a learned concept h is the probability of the instances incorrectly classified by h. That is,

$d(h,f) = \text{Prob}\{ x \in X: h(x) \neq f(x) \}$.

Prob{} is determined by an arbitrary sampling distribution, D, over X. Learning is accomplished by processing a learning procedure on a sample of instances called the training sample. Sampling is assumed to be with replacement with samples drawn independently.

For $0 < \varepsilon, \delta < 1$, a learning procedure is said to be a probably approximately correct (with respect to D) identification of the target concept

f if

$$\text{Prob}\{\ d(h,f) \geq \varepsilon\ \} \leq \delta.$$

Many algorithms are known for determining decision trees [Quinlan, 1986b; Cheng et al., 1988; Mingers, 1989a; Utgoff, 1989]. Recently, Ehrenfeucht and Haussler [1988] presented the first PAC learning algorithm for binary decision trees of rank at most r (rank is defined below). For any fixed rank r, the number of random examples and computation time required in this algorithm is polynomial in the number of attributes and linear in $1/\varepsilon$ and $\log(1/\delta)$.

We take the rank as a conciseness measure of a decision tree and give a pruning algorithm which gives the least upperbound of a pruning error.

We also determine the error introduced by pruning and give the required sample size to guarantee PAC-identification with accuracy parameter ε and confidence parameter δ.

III.Binary decision trees

1.Definitions

We begin with formal definitions of binary decision trees, their rank, the functions they represent and their error in an instance space. Our notation is similar to that given by Ehrenfeucht and Haussler [1988].

Definition III.0 *(A reduced binary decision tree)*: Let $V_n = \{v_1, \ldots, v_n\}$ be a set of n Boolean variables. Let $X_n = \{0,1\}^n$. A binary decision tree is defined as follows:

(i) If Q is a tree with only a root labelled either 0 or 1, then Q is a binary decision tree over V_n (Below we abbreviate this case by saying "Q=0" or "Q=1").

(ii) Let Q_0 be the left subtree of Q, and let Q_1 be the right subtree of Q. If the root node v of Q is in V_n and the left subtree Q_0 (a 0-Subtree), and right subtree Q_1 (a 1-Subtree) are

binary decision trees, then Q is also a binary decision tree.

We define an *internal node* i of a decision tree Q as a node in Q which has left and right subtrees. All nodes which are not internal nodes are called *leaf nodes* or simply leaves of a decision tree Q. We say that an internal node i is *informative* if it has only leaves and the leaves have different labels.

We say that a decision tree is *reduced* if each variable appears at most once in any path from the root to a leaf.

The *level* of node i is the number of predecessor nodes from the root. The height of a decision tree Q is defined as the maximum of the levels of all leaf nodes of Q.

A fully labeled tree is a tree for which every leaf has a 0 or 1 label.

A complete binary tree is a binary decision tree where every leaf is at the same level.

Definition III.1 *(A function and rank of a decision tree)*: A fully labeled binary decision tree Q represents a Boolean function f_Q defined as follows:

(i) If Q = 0 then f_Q is the constant function 0 and if Q = 1 then f_Q is the constant function 1.

(ii) Else if v_i is the label of the root of Q, then for any point $x=(a_1, ..,a_n) \in X_n$,

if $a_i=0$ then $f_Q(x) = f_{Q0}(x)$, else $f_Q(x)=f_{Q1}(x)$.

The rank of a decision tree Q, denoted r(Q), is defined as follows:

(i) If Q=0 or Q=1 then r(Q)=0.

(ii) Else if r_0 is the rank of the 0-subtree of Q and r_1 is the rank of the 1-subtree, then

$$r(Q) = \begin{cases} \max(r_0, r_1) & \text{if } r_0 \neq r_1 \\ r_0+1 (= r_1 + 1) & \text{otherwise} \end{cases}$$

Let T_n^r be the set of all binary decision trees over V_n of rank at most r and let F_n^r be the set of Boolean functions on X_n that are represented by trees in T_n^r.

The error of a decision tree Q is defined below.

Definition III.2: Let f be the target concept. The error of a decision tree Q, denoted e(Q) or e when Q is

understood, is defined as the probability of all x such that $f(x) \neq f_Q(x)$. That is,

$$e(Q) = \text{Prob}\ \{\ x : f(x) \neq f_Q(x)\ \}.$$

2.Finding Consistent Binary Decision Trees with Minimum Rank.

There are several algorithms for finding a decision tree using a training set S (see [Quinlan, 1986b, Cheng et al., 1988, Mingers, 1989a, Spangler et al., 1989 and Utgoff, 1989]). Here we focus on a PAC-learning method given by Ehrenfeucht and Haussler [1988].

Definition III.3: An example of a Boolean function f on X_n is a pair (x,f(x)), where $x \in X_n$. The example is positive if f(x)=1, else it is negative. A sample, S, of f is a set of examples of f. |S| denotes the number of examples in S. A decision tree Q over V_n is consistent with a sample S if for any example (x,f(x)) in S, $f(x)=f_Q(x)$. The rank of a sample S, denoted r(S), is the minimum rank of any decision tree that is consistent with S.

We say a variable v in V_n is informative (on S) if sample S contains at least one example in which the label of v is 0 and at least one example in which the label of v is 1. Let S_0^v be the set of all examples (x,f(x)) such that the label of v is 0, and let S_1^v be the set of all examples (x,f(x)) such that the label of v is 1.

The following two algorithms shown in Definition III.4 and III.5 can be used to determine a decision tree consistent with a sample S.

Definition III.4: The following procedure Find(S,k) is reproduced from Ehrenfeucht and Haussler [1988].

Procedure Find(S,k)

Input: A nonempty sample S of some Boolean function on X_n and an integer k, $n \geq k \geq 0$.

Output: A binary decision tree of rank

at most k that is consistent with S if one exists, else "none".

1. If all examples in S are positive, stop and return the decision tree Q = 1; if all examples are negative, stop and return Q = 0.

2. If k = 0, stop and return "none".

3. For each informative variable v in V_n

a. Let Q_0^v = Find(S_0^v,k-1) and Q_1^v = Find(S_1^v,k-1).

b. If both recursive calls are successful (i.e., neither Q_0^v = none, nor Q_1^v = none) then stop and return the decision tree with root labeled v, 0-subtree Q_0^v and 1-subtree Q_1^v.

c. If one recursive call is successful but the other is not, then

i) Re-execute the unsuccessful recursive call with rank bound k instead of

k-1, (i.e., if Q_1^v is a tree but Q_0^v = none then let Q_0^v = Find(S_0^v,k))

ii) If the re-executed call is now successful, then let Q be the decision tree

with root labeled v, 0-subtree Q_0^v and 1-subtree Q_1^v, else let Q = "none".

iii) Stop and return Q.

4. Stop and return "none".

Definition III.5: The following procedure Findmin(S) is reproduced from Ehrenfeucht and Haussler [1988].

Procedure Findmin(S)

Input: A nonempty sample S of some Boolean function on X_n.

Output: A minimal rank reduced binary decision tree of S.

1. Repeat Find(S,k) for k = 0,1,2,... until a decision tree is returned.

2. Stop and return Q.

Findmin()'s performance is given below.

Theorem III.6 *(A Decision tree with minimum rank: [Ehrenfeucht and Haussler, 1988]):* Given a sample S of a Boolean function on X_n, using Findmin(S) we can produce a decision tree that is consistent with S and has rank r(S) in time $O(|S|(n+1)^{2r(S)})$.

We will use Findmin() in our approach.

IV. Pruning a consistent decision tree to a desired rank

In this section we present a pruning algorithm that will be used with Findmin() to give certain theoretical results. We begin with definitions of pruning and labelling.

1.Definitions

There are many ways to prune a decision tree. We first address what we mean by pruning.

focus on deterministic methods for labelling. Later, we will look at two non-deterministic rules.

Definition IV.1 *(Deterministic Labelling):* Let i be an internal node in a decision tree Q, and Q(i) be a subtree of Q such that node i is the root of the tree Q(i).

1. Sample labelling:

Let s(i) be a subset of the sample S used to construct Q from its root to node i.

Let $s_0(i)$ denote the number of negative examples in s(i) and $s_1(i)$ denote the number of positive examples in s(i). If we prune Q at i,

then

(i) If $s_0(i) \geq s_1(i)$, label i as 0.

(ii) If $s_0(i) < s_1(i)$, label i as 1.

2. Tree labelling:

Let $l_1(i) = |\{x: f_Q(x)=1\}|$, where $x \in X_n$ such that $a_{i1},...,a_{ip}$ are the same. If we prune Q at i, then

(i) If $l_0(i) \geq l_1(i)$, label i as 0.

(ii) If $l_0(i) < l_1(i)$, label i as 1.

Below we give an illustrative example where the above two methods give a different labelling.

Consider the case of an instance space over two Boolean variables. Further suppose we have five training examples. In Table 1 below, n() denotes the number of examples for each point x. $f_Q()$ denotes the boolean function of the concept learned by constructing a consistent decision tree from these training examples.

Table 1: A training set of size five

x	n(x)	$f_Q(x)$
(1,1)	1	1
(1,0)	0	1
(0,1)	1	1
(0,0)	3	0

Suppose we prune Q at the root. If we label the root node by the sample labelling rule, the label of the root is 0 since $s_0()=3 \geq s_1()=2$. However, if we use the tree labelling rule, the label of the root will be 1 since $l_0()=1 < l_1()=3$.

2.Pruning

We now consider the following problem: Given a consistent decision tree with rank k, produce a pruned decision tree with rank r ($k > r$). We want a pruned tree to have the least amount of error due to pruning. We present an algorithm that solves this problem in time $O(2^{k+r}+|S|)$ using sample labelling and $O(2^{k+r}+(en/k)^k)$ using tree labelling.

For any decision tree Q of rank k>r, it is easily seen that one of the following mutually exclusive cases must hold.

Case 1: $r(Q_0) = k$ and $r(Q_1) < r$

Case 2: $r(Q_0) = k$ and $r \leq r(Q_1) < k$

Case 3: $r(Q_0) < r$ and $r(Q_1) = k$

Case 4: $r \leq r(Q_0) < k$ and $r(Q_1) = k$

Case 5: $r(Q_0) = r(Q_1) = k-1$

For a given decision tree of rank

k, there may be many ways to prune it to rank r. Since our objective is to minimize the pruning error, one possible strategy is to prune the least number of nodes of a consistent decision tree Q. In other words, we will prune Q to the largest subtree having the desired rank. The idea of our pruning algorithm is as follows. In Cases 1 and 3, pruning only one subtree with rank k to rank r is enough to form a pruned decision tree with rank r. Pruning the other subtree with rank less than r to a lower rank is not needed since the resulting tree will still have same rank. Similarly, using this minimum node pruning rule we have two possible alternative ways of pruning in Cases 2 and 4.

(i) Prune one subtree with rank k to rank r-1 and prune the other subtree to rank r.

(ii) Prune one subtree with rank k to rank r and prune the other subtree to rank r-1.

In the algorithm shown below, we use method (i) and in Section V, we show that this method gives the least pruning error.

Finally, in Case 5, we will arbitrarily prune the tree by giving the preference to the 0-subtree.

3.A pruning algorithm

Below is pruning algorithm Prune( ). This algorithm can prune any binary tree. However, we restrict our initial input to trees produced by Findmin(S) in order to later guarantee PAC identification. In the following Q $\Leftarrow$ P means Q is replaced by P.

Procedure Prune(r,k,Q,S)

Input: A decision tree Q with rank at most k, a sample S and an integer r such that $0 \le r \le k$.

Output : A decision tree with rank at most r.

Let $s_0(Q,S)$ = the number of examples in S such that $f_Q(x)=0$.

Let $s_1(Q,S)$ = the number of examples in S such that $f_Q(x)=1$.

Let $l_0(Q)$ = the number of instances x $\in X_n$ such that $f_Q(x)=0$.

Let $l_1(Q)$ = the number of instances $x \in X_n$ such that $f_Q(x)=1$.

1. If $r(Q) \le r$, then stop and return Q.

2. If r=0, then stop and label Q by the appropriate labelling rule:

   1) Sample labelling:

      $Q=0$ if $s_0(Q,S) \ge s_1(Q,S)$,

      otherwise $Q=1$.

   2) Tree labelling:

      $Q=0$ if $l_0(Q) \ge l_1(Q)$,

      otherwise $Q=1$.

      Return Q.

3. Let $k_0 = r(Q_0)$, $k_1 = r(Q_1)$.

   Let S_0 be the set of all examples (x,f(x)) in S such that $x=(a_1,...,a_n)$ and $a_i=0$,

   where $v_i \in V_n$ is the root of Q.

   Let S_1 be the set of all examples (x,f(x)) in S such that $x=(a_1,...,a_n)$ and $a_i=1$,

   where $v_i \in V_n$ is the root of Q.

   Case 1: If $k_0 = k$ and $k_1 < r$

   then $Q_0 \leftarrow \text{Prune}(r,k_0,Q_0,S_0)$.

   Case 2: If $k_0 = k$ and $r \le k_1 < k$

   then $Q_1 \leftarrow \text{Prune}(r,k_1,Q_1,S_1)$,

   $Q_0 \leftarrow \text{Prune}(r-1,k_0,Q_0,S_0)$

   Case 3: If $k_0 < r$ and $k_1 = k$

   then $Q_1 \leftarrow \text{Prune}(r,k_1,Q_1,S_1)$.

   Case 4: If $r \le k_0 < k$ and $k_1 = k$

   then $Q_0 \leftarrow \text{Prune}(r,k_0,Q_0,S_0)$,

   $Q_1 \leftarrow \text{Prune}(r-1,k_1,Q_1,S_1)$

   Case 5: If $k_0 = k_1 = k-1$

   then $Q_0 \leftarrow \text{Prune}(r,k_0,Q_0,S_0)$,

   $Q_1 \leftarrow \text{Prune}(r-1,k_1,Q_1,S_1)$

4. Stop and return Q.

To obtain our theoretical results it will be useful to have another pruning algorithm. This alternative pruning method prunes one subtree of rank k to rank r and prunes the other subtree to rank r-1. We define W-Prune(), as this alternative pruning algorithm.

Procedure W-Prune(r,k,Q,S)

Input: A decision tree Q with rank at most k, a sample S

and an integer r such that $0 \le r \le k$.

Output : A decision tree with rank at most r.

Let $s_0(Q,S)$ = the number of examples in S such that $f_Q(x)=0$.

Let $s_1(Q,S)$ = the number of examples in S such that $f_Q(x)=1$.

Let $l_0(Q)$ = the number of instances x $\in X_n$ such that $f_Q(x)=0$.

Let $l_1(Q)$ = the number of instances x $\in X_n$ such that $f_Q(x)=1$.

1. If $r(Q) \le r$, then stop and return Q.

2. If r=0, then stop and label Q by the appropriate labelling rule:

   1) Sample labelling:

      Q=0 if $s_0(Q,S) \ge s_1(Q,S)$, otherwise Q=1.

   2) Tree labelling:

      Q=0 if $l_0(Q) \ge l_1(Q)$,

      otherwise Q=1.

      Return Q.

3. Let $k_0 = r(Q_0)$, $k_1 = r(Q_1)$.

   Let S_0 be the set of all examples (x,f(x)) in S such that $x=(a_1,...,a_n)$ and $a_i=0$,

   where $v_i \in V_n$ is the root of Q.

   Let S_1 be the set of all examples (x,f(x)) in S such that $x=(a_1,...,a_n)$ and $a_i=1$,

   where $v_i \in V_n$ is the root of Q.

   Case 1: If $k_0 = k$ and $k_1 < r$

   then $Q_0 \leftarrow$ W-Prune(r,k_0,Q_0,S_0).

   Case 2: If $k_0 = k$ and $r \le k_1 < k$

   then $Q_1 \leftarrow$ W-Prune$(r-1,k_1,Q_1,S_1)$,

   $Q_0 \leftarrow$ W-Prune(r,k_0,Q_0,S_0)

   Case 3: If $k_0 < r$ and $k_1 = k$

   then $Q_1 \leftarrow$ W-Prune (r,k_1,Q_1,S_1).

   Case 4: If $r \le k_0 < k$ and $k_1 = k$

   then $Q_0 \leftarrow$ W-Prune$(r-1,k_0,Q_0,S_0)$,

   $Q_1 \leftarrow$ W-Prune(r,k_1,Q_1,S_1)

   Case 5: If $k_0 = k_1 = k-1$

   then $Q_0 \leftarrow$ W-Prune(r,k_0,Q_0,S_0),

   $Q_1 \leftarrow$ W-Prune$(r-1,k_1,Q_1,S_1)$

4. Stop and return Q.

The following Lemma IV.2 shows that Prune(r,k,Q,S) and W-Prune(r,k,Q,S) always give a pruned tree with the desired rank at most r.

Lemma IV.2: The procedures Prune (r,k,Q,S) and W-Prune(r,k,Q,S) are correct.

Proof: The proof is by induction on r. If r=0 and k≥0 then Prune() and W-Prune() return Q=0 or Q=1. Hence, r(Q) = 0 = r.

Now suppose Prune() and W-Prune() run correctly on all cases requiring rank r′ ≤ r-1 and k ≥ 0. In each of the five cases treated within Prune() and W-Prune(), either a tree of rank less than or equal to r-1 is to be produced, or a tree of rank at most r is to be produced. By assumption, for those requiring rank less than or equal to r-1, the algorithm will work correctly. So we turn our attention to the case requiring rank at most r.

Without loss of generality, suppose Q_0 requires rank at most r. Prune(r,k_0,Q_0,S_0) and W-Prune(r,k_0,Q_0, S_0) will operate on a tree with one fewer variable than Q. Since the tree is reduced, the number of variables used in Q_0 from its root to leaves must be less than or equal to n-1. Since only one subprocedure requires a tree with rank at most r in each recursion, we divide that subprocedure into subprocedures until the subprocedure requiring rank at most r will be operated on a tree with rank less than or equal to r. So, ultimately in the recursion, all subprocedures except one require trees with rank less than or equal to r-1, and the subprocedure requiring a tree with rank at most r will be operated on a tree with rank less than or equal to r since the number of variables used in the tree is reduced enough to guarantee that the rank of the tree is less than or equal to r. If r(Q) ≤ r, by Step 1, the Prune() and W-Prune() return a tree with rank at most r correctly. Since Prune() and W-Prune() are correct for all other subprocedures requiring rank less than or equal to r-1, by induction, the procedure Prune(r,k,Q,S) and W-Prune(r,k,Q,S) return a pruned tree with rank at most r.

Therefore, the procedure Prune(r,k, Q,S) and W-Prune(r,k,Q,S) are correct for all 0≤r≤k. ■

The following Lemma IV.3 will be used to prove the time complexity of the procedure Prune().

Lemma IV.3 *(An upperbound on the*

number of nodes in a reduced decision tree over V_n: [Ehrenfeucht and Haussler, 1988]): Let j be the number of nodes in a reduced decision tree over V_n of rank k, where $n \geq k \geq 1$. Then

$$j < 2(en/k)^k,$$

where e is the base of the natural logarithm.

Lemma IV.4: For any nonempty tree Q with rank $k \geq 0$, the time of Prune(r,k,Q,S) is $O(2^{k+r}+|S|)$ for sample labelling, and $O(2^{k+r}+(en/k)^k)$ for tree labelling.

Proof: Prune() will stop either when r=0 or when k reduces to r. For each call of Prune(), r or k will be reduced by at least one and at most two new recursive calls of Prune() are made. Since we do not have subprocedures if $r+k \leq 2$, there are at most r+k-2 steps. So, the total number of calls of Prune() will be at most 2^{k+r-1}. For each call of Prune(), there are a constant number of unit operations (mainly, comparisons) except for the labelling time. The time for the labelling step (Step 2) depends on the labelling rule. If we label a pruned node using sample labelling, at most $|S|$ additional time is needed since S_0 and S_1 in the Prune() are disjoint subsets of S and labelling occurs at most once in each subprocedure. If we label a pruned node using tree labelling, at most $(en/k)^k$ additional time is needed since Q_0 and Q_1 are disjoint subtrees of Q and labelling occurs at most once in each subprocedure. Also, the number of leaf nodes which we need to scan to label the node is less than or equal to $(en/k)^k$ by Lemma IV.3. So, the time of Prune(r,k,Q,S) is $O(2^{k+r}+|S|)$ for sample labelling, and$O(2^{k+r}+(en/k)^k)$ for tree labelling.■

Lemma IV.4 gives the time complexity of this algorithm. If we use the sample labelling rule, the time required for pruning is independent of the number of variables, n, and far less than the tree construction time of Findmin() in Theorem III.6 since $n \geq 1$, $k \geq r$ and $|S| \geq 1$. If we use the

tree labelling rule, the time required for pruning is polynomial in n for fixed k and less than the time of Findmin() since $|S| \geq 1$.

V.Determining the pruning error

To determine a PAC learning result for pruning, we need to bound the pruning error. Here we give a formal definition of the pruning error and the least upperbound of the pruning error for a given rank tree. We can define the pruning error over the training sample or over the whole instance space. Here we use the whole instance space. The pruning error over the training sample will always be positive since we are assuming that the starting decision tree is reduced and is consistent with the training sample. However, if we define the pruning error over the instance space, we may get different results since the pruned tree may perform better than the unpruned tree as shown in many studies [Quinlan, 1987a; Mingers, 1989b].

Definition V.0: Given a decision tree P(Q) formed by pruning a decision tree Q produced by Findmin(), the pruning error e(Q,P(Q)), is defined as the difference between the error of P(Q) and the error of Q.

$$e(Q,P(Q)) = e(P(Q)) - e(Q)$$
$$= \text{Prob}\{ x : f(x) \neq f_{P(Q)}(x) \} - \text{Prob}\{ x : f(x) \neq f_Q(x) \}.$$

For an arbitrary probability distribution D and any decision tree Q in which all nodes are informative, we can easily verify that the pruning error is in the range (-1,1) since, by definition, the pruning error is the difference of two probabilities, where e(Q) is in the range [0,1) and e(P(Q)) is in the range (0,1). Since we assume that all examples are drawn according to D without error and Findmin(S) guarantees that all internal nodes having only leaves in Q are informative, e(P(Q)) must be positive. e(P(Q)) cannot be 1 since P(Q) cannot

be the null tree. So the pruning error cannot reach boundary points -1 or 1. Below we show that both positive and negative cases of e(Q,P(Q)) can be realized.

1.Possible cases of the pruning error

Consider the case of an instance space over three Boolean variables. Further suppose we have a target concept f() and probability distribution D() as shown in Tables 2 and 3 below. Suppose we take 10 training examples under D(). In Table 2 and 3 below, n() denotes the number of examples for each point x. $f_Q()$ denotes the boolean function of the concept learned by constructing a consistent decision tree from these training examples. By pruning the decision tree Q, we get a pruned decision tree P(Q). $f_{P(Q)}()$ denotes the boolean function of the concept learned from the pruned decision tree. We label each pruned node by the sample labelling rule.

Table 2: Positive error case

x	f(x)	D(x)	n(x)	$f_Q(x)$	$f_{P(Q)}(x)$
(1,1,1)	0	.001	0	1	1
(1,1,0)	1	.001	3	1	1
(1,0,1)	1	.001	0	0	0
(1,0,0)	0	.001	2	0	0
(0,1,1)	0	.001	2	0	0
(0,1,0)	1	.001	0	0	0
(0,0,1)	1	.993	2	1	0
(0,0,0)	0	.001	1	0	0

In this decision tree, r(Q) = 2 and r(P(Q)) = 1.

The pruning error for this case is

Prob { x : $f(x) \neq f_{P(Q)}(x)$ } - Prob { x : $f(x) \neq f_Q(x)$ }

= (.001 + .001 + .001 + .993) - (.001 + .001 + .001) = .993

Table 3: Negative error case

x	f(x)	D(x)	n(x)	$f_Q(x)$	$f_{P(Q)}(x)$
(1,1,1)	0	.001	0	1	1
(1,1,0)	1	.001	3	1	1
(1,0,1)	1	.001	0	0	0
(1,0,0)	0	.001	2	0	0
(0,1,1)	0	.001	1	0	1

(0,1,0)	1	.993	0	0	1
(0,0,1)	1	.001	3	1	1
(0,0,0)	0	.001	1	0	1

In this decision tree, r(Q) = 2 and r(P(Q)) = 1.

The pruning error for this case is

Prob { x : f(x) ≠ $f_{P(Q)}$(x) } - Prob { x : f(x) ≠ f_Q(x) }

= (.001 + .001 + .001 + .001) - (.001 + .001 + .993) = -.991

2.The least upperbound of the pruning error

Here we develop a series of lemmas which we need to determine the least upperbound of the pruning error. This is used to guarantee PAC identification of a learning algorithm. We begin with a formal definition of the least upperbound of the pruning error.

Let W_n^r denote the set of all fully-labeled, reduced, binary trees of rank r over V_n, in which all variables are informative and all internal nodes which have only leaves are informative. Since we are using Findmin() which finds a decision tree $Q \in W_n^r$, we need only consider $Q \in W_n^r$. However, much of the following analysis can be directly applied to other situations.

Definition V.1: The least upperbound of the pruning error, denoted $\eta_{k,r}$, for given rank k, and target r is defined as follows:

Let $\eta_{k,r} = \sup_{Q \in W_n^k} \inf_{P \in \rho(Q)} e(Q,P)$.

For any $Q \in W_n^k$, $\rho(Q)$ is the set of all subtrees of Q of rank r pruned from Q.

For convenience, we define $\eta_r \equiv \eta_{r,r-1}$.

From earlier discussions, we know that $\eta_{k,r}$ can be equal to 1 for an arbitrary distribution D. Since the population distribution is usually unknown for real world situations, it is difficult to characterize $\eta_{k,r}$ further. So, without some restrictions on distribution D, we cannot improve or characterize the upperbound of the pruning error. However, for totally unknown population distributions, we may assume an equally likely distribution (i.e., a uniform distribution), or we may assume that thesampling

distribution is equal to the population distribution. Here we start with the assumption of an equally likely distribution D to determine the error of pruning. The pruning error also depends on the labelling rule. Here we use the tree labelling rule. As will be shown, these assumptions will guarantee that the least upperbound of the pruning error is positive and less than or equal to 0.5. In the following we give several results characterizing $\eta_{k,r}$ under these assumptions.

To facilitate our effort to determine $\eta_{k,r}$, we will need to define a Maximal tree. Let Q(j) be a subtree of Q with an internal node j of Q as its root node.

Definition V.2: A Maximal tree of a reduced decision tree Q with rank r for given n, $n \geq r$, is defined as follows:

1) If n = r, then Q is a complete binary tree with height n.

2) If n > r, then for each internal node j,

i) if the level of node j is less than n - r(Q(j)), then one of the subtrees of Q(j) has rank r(Q(j)) and the other subtree has rank r(Q(j))-1; or

ii) if level of node j = n - r(Q(j)), then Q(j) is a complete binary tree with height r(Q(j)).

Let M_n^r denote the set of all maximal trees of rank r over V_n.

Based on the above definition, it can be easily seen that the number of nodes are equal for all maximal trees of the same rank for given n, and the number of nodes in a maximal tree increases as the rank increases since a maximal tree $Q \in M_n^r$ can always be found by deleting at least one node from a higher rank maximal tree $Q' \in M_n^{r+1}$.

Under the assumptions of a uniform distribution and the tree labelling rule, Lemma V.3 and Lemma V.4 characterizes $\eta_{k,r}$.

Lemma V.3: When D is a uniform distribution and Prune() uses the tree labelling rule and is given $Q \in W_n^k$ as input, the least upperbound of the pruning error, $\eta_{k,r}$, is strictly positive and less than 0.5 when r>0. $\eta_{k,r} = 0.5$ only when r=0.

Proof: Let $Q \in W_n^k$. Suppose some internal node "i" in Q is at level m ($m<n$). Further suppose that we prune Q to P(Q) at node "i" leaving "i" a leaf node in P(Q). Let p("i") be the probability of instances covered by the leaves of the subtree Q(i) in Q. Under a uniform distribution, $p("i") = 2^{n-m}/2^n = (0.5)^m$. Let e("i") be the probability of incorrect classifications in the instances covered by the leaves of subtree Q(i) over the whole instance space.

By the definition of $l_0(i)$ and $l_1(i)$ in Definition IV.1, $l_0(i)+l_1(i) = 2^{n-m}$. Suppose $\alpha_0(i)$ is the number of incorrect classifications of $l_0(i)$ and $\alpha_1(i)$ is the number of incorrect classifications of $l_1(i)$. Then clearly

$$e("i") = \alpha_0(i)/2^n + \alpha_1(i)/2^n$$

where $0 \le \alpha_0(i) \le l_0(i)$ and $0 \le \alpha_1(i) \le l_1(i)$.

If we prune Q at node "i", then in P(Q) the label of the leaf node "i" is either 0 or 1. If we label the node "i" by the "tree labelling" rule, then node "i" will have the label which covers the larger number of instances. If $l_0(i) \ge l_1(i)$, then the label of the node i is 0 and

$$e("i") = \alpha_0(i)/2^n + l_1(i)/2^n - \alpha_1(i)/2^n.$$

By the definition of the pruning error,

$$e(Q,P(Q)) = e(P(Q)) - e(Q)$$
$$= [\ \alpha_0(i)/2^n + l_1(i)/2^n - \alpha_1(i)/2^n\ ] - [\ \alpha_0(i)/2^n + \alpha_1(i)/2^n\ ]$$
$$= l_1(i)/2^n - 2\alpha_1(i)/2^n \le l_1(i)/2^n \le (1/2)(0.5)^m$$

since $l_1(i) \le 2^{n-m-1}$.

Similarly, if $l_0(i) < l_1(i)$, then the label of the node i is 1 and

$$e("i") = \alpha_1(i)/2^n + l_0(i)/2^n - \alpha_0(i)/2^n.$$

$$e(Q,P) = l_0(i)/2^n - 2\alpha_0(i)/2^n \le l_0(i)/2^n < (1/2)(0.5)^m$$

since $l_0(i) < 2^{n-m-1}$.

In both cases, the pruning error does not exceed half of p("i"). In other words, the leaf node "i" in P(Q) correctly classifies at least half of the corresponding instances. Since this is true for all pruned nodes, the pruned tree P(Q) correctly classify at least a half of total instances. Therefore, the pruning error $e(Q,P(Q)) \le 0.5$.

Further, $l_0(i)$ and $l_1(i)$ are non-zero for any nodes in $Q \in W_n^k$ since all

nodes in Q are informative. Hence, $e(Q,P(Q)) > 0$ and $\eta_{k,r} > 0$ for all $k>r\geq 0$.

If $r>0$, at least one subprocedure of Prune() stops at step 1 since Prune() reduces k only in one of its subprocedures and eventually k reduces to less than or equal to r. In that subprocedure, Prune() does not prune that subtree of Q. So the probability over the instance space for which pruning is needed in Q is strictly less than 1. Hence, $e(Q,P(Q)) < 0.5$ and $\eta_{k,r} < 0.5$.

If $r=0$, then the probability over the instance space for which pruning is needed in Q is equal to 1. Hence, $e(Q,P(Q)) \leq 0.5$ and $\eta_{k,r} = 0.5$ can only be realized when the probability of all the negative examples and that of all the positive examples are equal, and when Q has no error over the instance space, for all $k>r=0$.

■

Lemma V.4: Let $Q \in W_n^k$ and $P \in \rho(Q)$. The following are true.

a) The pruning error, $e(Q,P)$, is a non-decreasing function of the amount of pruning, k-r.

b) $\eta_{k,r}$ is attained at a maximal tree P(Q) in M_n^r.

c) $\eta_{k,r}$ is a non-increasing function of r.

Proof: a) Suppose that an internal node "a" in Q is at level m ($m<n$) with left child node "b" and right child node "c" each at level m+1. Under a uniform distribution, the probability of instances covered by node "a" at level m is $(0.5)^m$. Let e("i") be the probability of an incorrect classificati-
on in the instances covered by the leaves of the subtree Q(i).

Suppose P(Q) and P'(Q) are two pruned trees of Q, where P(Q) is found by pruning at nodes "b" and "c" of Q and P'(Q) is found by pruning at node "a" of Q. Further suppose $e("b") = \alpha$ and $e("c") = \gamma$ at level m+1. Using the same arguments as in the proof of Lemma V.3, $0 \leq \alpha,\gamma \leq (0.5)^{m+2}$. Then

$$e("a") = e("b") + e("c") = \alpha + \gamma$$

in P(Q).

But if we prune Q at node "a", the label of the leaf node "a" is either 0 or 1. If the label of "a" is 0, then in P'(Q),

$$e("a") = \alpha + ((0.5)^{m-1} - \gamma)$$
$$= (0.5)^{m-1} + \alpha - \gamma.$$

So, [e("a") in P'(Q)] - [e("a") in P(Q)]

$$= ((0.5)^{m-1} + \alpha - \gamma) - (\alpha + \gamma)$$
$$= (0.5)^{m-1} - 2\gamma \geq 0$$

since $\gamma \leq (0.5)^{m-2}$.

Similarly, if the label of "a" is 1,

[e("a") in P'(Q)] - [e("a") in P(Q)]

$$= ((0.5)^{m-1} + \gamma - \alpha) - (\alpha + \gamma)$$
$$= (0.5)^{m-1} - 2\alpha \geq 0$$

since $\alpha \leq (0.5)^{m-2}$.

So, the error of the subtree at node "a" in P'(Q) is greater than or equal to that of P(Q). Since all other nodes are equal except node "a" in P(Q) and P'(Q), $e(P'(Q)) \geq e(P(Q))$. Hence, since e(Q) is equal in both cases, the pruning error, e(Q,P), is a non-decreasing function of the amount of pruning.

b) By the definition of a maximal tree, for given n and r, any non-maximal tree has less nodes than a maximal tree of equal rank. To find a non-maximal tree of desired rank by pruning Q, we need to prune more nodes than the amount needed when we find a comparable maximal tree of equal rank. So, the least amount of pruning occurs when we prune Q to a maximal tree P(Q) of rank r. From (a), the least upperbound of the pruning error, $\eta_{k,r}$, is attained at a maximal tree P(Q) in M_n^r.

c) From (b), $\eta_{k,r}$ occurs at a maximal tree. Since a maximal tree with rank r+1 has more nodes than a maximal tree with rank r for a given number of attributes n, we need to prune more nodes when we prune a tree with rank k to a tree with rank r than to a tree with rank r+1. So, $\eta_{k,r-1} \leq \eta_{k,r}$ for all r.

■

Lemma V.5 *(Characterization of $\eta_{k,r}$)*:

$\eta_{k,0} = 0.5$ for $k>0$.

$\eta_{k,1} = 0.5\ (\ 1 - (0.5)^{k-1})$ for $k>1$.

$\eta_{k,2} = 0.5\ (\ 1 - (0.5)^{k-2}) - 2(k-2)(0.5)^{k-2}$ for $k>2$.

$\eta_{k,3} = 0.5 (1 - (0.5)^{k-3}) - (k-3)(k+8)(0.5)^{k-3}$ for $k>3$.

$\eta_{k,4} = 0.5 (1 - (0.5)^{k-4}) - (k-4)\{(k^2+13k+84)/3\}(0.5)^{k-4}$ for $k>4$.

$\eta_{k,r} = \sum_{m=r-1}^{k} (0.5)^{k-m+1} \eta_{m,r-1}$, $k>r$.

$\leq 0.5 - \{ 1 + (k-r)(0.5)^2\}(0.5)^{k-r-1}$ for $k>r>1$.

$\eta_1 = 1/2 = 0.5$. $\eta_2 = 1/4 = 0.25$. $\eta_3 = 3/16 = 0.1875$.

$\eta_4 = 5/32 = 0.156$. $\eta_5 = 35/256 = 0.1367$. $\eta_6 = 126/1024 = 0.123$.

$\eta_7 = 462/4096 = 0.1128$.

$\eta_8 = 1716/16384 = 0.1047$.

Proof: First we prove that $\eta_{k,1} = 0.5 (1 - (0.5)^{k-1})$. To reduce the rank of a tree to 1, the procedure Prune() will prune each subtree with rank greater than one at each level from level 1 to level k-1. So, by Lemma V.3, the least upperbound of the pruning error for an equally likely distribution will be

$$\eta_{k,1} = (0.5)^2 + (0.5)^3 + \ldots + (0.5)^k = (0.5) (1 - (0.5)^{k-1}).$$

Now, we prove the recursive relation. With Prune(r,k,Q,S), pruning a maximal tree with rank k to a tree with rank r will result in calls to two subprocedures Prune(r-1,k,Q,S) and Prune(r,k-1,Q,S). As the level of pruning goes down by one level, $\eta_{k,r}$ of the lower level reduces to a half of that of the higher level. So, the pruning errors will have the following relationship.

$$\eta_{k,r} = (0.5)\eta_{k,r-1} + (0.5)\eta_{k-1,r}.$$

If we continue branching the procedure Prune(r,k,Q,S) at each level, we will get

$$\eta_{k,1} = (0.5)\eta_{k,r-1} + (0.5)\{(0.5)\eta_{k-1,r-1} + (0.5)\eta_{k-2,r}\}$$

$$= (0.5)\eta_{k,r-1} + (0.5)^2\eta_{k-1,r-1} + \ldots + (0.5)^i\eta_{k-i-1,r-1} + \ldots + (0.5)^{k-2}\{(0.5)\eta_{r+2,r-1} + (0.5)\eta_{r-1,r}\}.$$

Since $\eta_{r-1,r} = (0.5)\eta_{r-1,r-1}$ in a maximal tree,

$$\eta_{k,r} = \sum_{m=r-1}^{k} (0.5)^{k-m+1} \eta_{m,r-1} \qquad k>r$$

where $r+1 \leq m \leq k$.

Here we use mathematical induction to show the upperbound of $\eta_{k,r}$.

For r=2, the above bound holds because

$\eta_{k,2} = 0.5 - (0.5)^{k-2} - 2(k-2)(0.5)^{k-2}$

$= 0.5 - (0.5)^{k-3} - (k-2)(0.5)^{k-1}$

$= 0.5 - \{1 + (k-2)(0.5)^2\}(0.5)^{k-1}$ for $k>r>1$.

Suppose it is true for r=p-1, where p≥3. Then

$$\eta_{k,p} = \sum_{m=p-1}^{k} (0.5)^{k-m-1} \eta_{m,p-1}$$

$$= \sum_{m=p-1}^{k} (0.5)^{k-m-1} \{ 0.5 - (0.5)^{m-p-2} - (m-p+1)(0.5)^{m-p-4} \}$$

$$= \sum_{m=p-1}^{k} \{ (0.5)^{k-m-2} - (0.5)^{k-p-3} - (m-p+1)(0.5)^{k-p-4} \}$$

$$< 0.5 - (0.5)^{k-p-1} - (k-p)(0.5)^{k-p-3}.$$

Thus it holds for r=p. Therefore,

$\eta_{k,r} \le 0.5 - \{1 + (k-r)(0.5)^2\}(0.5)^{k-r-1}$ for $n>r>1$. ■

Lemma V.6: Let $\eta_{k,r}'$ be the least upperbound of the pruning error using algorithm W-Prune() and the tree labelling rule. When D is a uniform distribution, $\eta_{k,r} \le \eta_{k,r}'$ for all $k>r\ge 0$.

Proof: In the Lemma V.3 we assumed only that D is a uniform distribution and the pruning algorithm is the tree labelling rule. Hence, it holds for the procedure W-Prune() also. So, when D is a uniform distribution, $\eta_{k,0} = \eta_{k,0}' = 0.5$ for all $k>0$.

Now we show that $\eta_{k,1}' = 0.5(1-(0.5)^{n-1})$. To reduce the rank of a tree to 1, the procedure W-Prune() will prune each subtree with rank greater than one at each level from level 1 to level n-1. So, by Lemma V.3, the least upperbound of the pruning error for an equally likely distribution will be

$$\eta_{k,1}' = (0.5)^2 + (0.5)^3 + \ldots + (0.5)^n$$

$$= (0.5)(1 - (0.5)^{n-1}).$$

Since $k\le n$, $\eta_{k,1} \le \eta_{k,1}'$ for all $k>1$. Notice that $\eta_{k,1}'$ is independent of k. So, $\eta_{k,1}' = 0.5(1-(0.5)^{n-1})$ for all $1<k\le n$.

Now we show the resursive relation. With W-Prune(r,k,Q,S), pruning a maximal tree with rank k to a tree with rank r will result in calls to two subprocedures W-Prune(r,k,Q,S) and W-Prune(r-1,k-1,Q,S). As the level of pruning goes down by one

level, $\eta_{k,r}$ of the lower level reduces to a half of that of the higher level. So, the pruning errors will have the following relationship:

$$\eta_{k,r}' = (0.5)\eta_{k,r}' + (0.5)\eta_{k-1,r-1}'.$$

This can be simplified to

$$\eta_{k,r}' = \eta_{k-1,r-1}'.$$

So, the following relationship holds:

$$\eta_{k,r}' = \eta_{k-1,r-1}' = \eta_{k-2,r-2}' = \cdots = \eta_{k-(r-1),1}'.$$

Since $\eta_{k,r}$ is a non-increasing function of r by Lemma V.4 (c),

$\eta_{k,r} \le \eta_{k,1} \le \eta_{k-(r-1),1}' = \eta_{k,r}'$ for all $k>r\ge1$ follows.

So, when D is a uniform distribution, $\eta_{k,r} \le \eta_{k,r}'$ for all $k>r\ge0$. ■

Theorem V.7: Under the assumption of an equally likely distribution over the instance space, procedure Prune(r,k,Q,S) with the tree labelling rule gives the least upperbound of the prunning error $\eta_{k,r}$ in time $O(2^{k+r}+(en/k)^k)$.

Proof: Case 1 and 3: By reducing the rank of the subtree with rank k to r, the rank of the whole tree will be r. Since $\eta_{k,r}$ is a non-decreasing function of r for fixed k by Lemma V.4 (c), it gives the least upperbound of the pruning error.

Case 2 and 4: There are only two alternative pruning ways which may give the least upperbound error.

1) prune one subtree with rank k to r-1 and then prune the other subtree to rank r; and

2) prune one subtree with rank k to r and then prune the other subtree to r-1.

By Lemma V.6, $\eta_{k,r} \le \eta_{k,r}'$. So, 1) gives the least upperbound error.

Case 5: Since $\eta_r>0$ for all r by Lemma V.3, pruning one subtree with rank k-1 to a tree with rank r and the other subtree to rank r-1 gives the least upperbound of the pruning error. ■

Lemma V.8 *(An upperbound of $\eta_{k,r}$):*

Let $\mu_{n,r} = 0.5 - (0.5)^{n-r+1}$ for $n\ge r=1$

$= 0.5 - \{1 + (n-r)(0.5)^2\}(0.5)^{n-r+1}$ for $n\ge r>1$.

Then $\eta_{k,r} \le \mu_{n,r} < 0.5$ for $0<r\le k\le n$.

Proof: In the proof of Lemma V.5 we have shown that

$$\eta_{k,r} = (0.5)\eta_{k,r-1} + (0.5)\eta_{k-1,r}.$$

Since $\eta_{k,r} \le \eta_{k,r-1}$ by Lemma V.4 (c), $\eta_{k,r} \ge \eta_{k-1,r}$. So, $\eta_{k,r}$ is a non-decreasing function of k. And since $r(Q)=k \le n$ for all reduced binary decision trees, Q, over V_n, $\eta_{k,r} \le \eta_{n,r}$. By Lemma V.5, for

$r=1$, $\eta_{n,r} = \mu_{n,r}$ and for

$r>1$, $\eta_{n,r} \le \mu_{n,r} = 0.5 - \{ 1 + (n-r)(0.5)^2\}(0.5)^{n-r+1}$ for $n \ge r > 1$.

For $n-r \ge 0$, the latter term is positive. So, $\mu_{n,r}$ is strictly less than 0.5.

■

VI. Sample size required for PAC identification.

We consider the size of a sample sufficient for PAC identification with pruning in finite domains. Blumer et al. [1987] have given a lower bound of the sample size required for PAC identification of any consistent learning algorithm. They showed that for N given finite rules in the hypothesis space, any rule agreeing with $m = (1/\varepsilon)\ln(N/\delta)$ or more randomly chosen examples has error greater than ε only with probability less than δ. This model assumes that there exists a rule in the hypothesis which correctly classifies all the training examples.

The number of possible rules, N, is used as a measure of the complexity of the hypothesis space in PAC-learning. We will use the following lemma when we quantify the hypothesis space for binary decision trees.

Recall that T_n^r denotes the set of all binary decision trees over V_n of rank at most r and F_n^r denotes the set of Boolean functions on X_n that are represented by trees in T_n^r. Then $|F_n^r|$ gives the number of rules in the hypothesis space of decision trees of rank at most r.

Lemma VI.0 *(An upperbound on* $|$

$F_n^r|$:*[Ehrenfeucht and Haussler, 1988]):*

$|F_n^r| = 2$, for $r = 0$.

$|F_n^r| \le (8n)^{(en/r)}$, for $n>r>0$.

$|F_n^r| = 2^2$, for $n = r$.

Consider now what happens if we prune a consistent decision tree. Since pruning a consistent decision tree built with informative variables introduces error over the training samples, the pruned tree P is not consistent with all the training samples. Angluin and Laird [1988] introduced into PAC-learning a model of random errors, or "noise", in the sampling. They assumed that it is possible to draw examples from the relevant distribution D without error, but that the process of determining and reporting whether the example is positive or negative is subject to independent random mistakes with some unknown probability $\eta < 0.5$. They called this process "the classification noise process" and developed a result which guarantees PAC-identification for the process. Their result is as folllows.

Lemma VI.1 *[Angluin and Laird, 1988]:*

For any finite set of N rules,
if we draw a sequence of

$$m \ge [2/\{\varepsilon^2(1-2\eta_b)^2\}]\ln(2N/\delta)$$

random examples for target concept f from an arbitrary probability distribution D with classification noise η (an upperbound is $\eta_b < 1/2$) and find any hypothesis h that minimizes the probability of misclassification in the examples, then

$$\text{Prob}\{\ d(h,f) \ge \varepsilon\ \} \le \delta.$$

Our model does not assume classification noise. However, the pruned tree is known to be in an hypothesis space where no function h is exactly equivalent to the target function f. Hence, the concepts we learn with pruning appear as if they are subject to a classification error.

Since we defined the pruning error

over the instance space, the pruning error can be considered as a classification noise in the random examples.

A direct application of Lemma VI.1 with our results gives our main result.

Theorem VI.2: For any $n \geq r > 0$, any target function $f \in F_n^p$, $p \leq n$, an equally likely probability distribution D on X_n and any $0 < \varepsilon, \delta < 1$, given a sample S derived from a sequence of

$$m \geq [2/\{\varepsilon^2(1-2\mu_{n,r})^2\}]\{(en/r)^r \ln(8n)+\ln(2/\delta)\}$$

random examples of f chosen independently according to D, with probability $1-\delta$, Findmin(S) and Prune(r,k,Q,S) using tree labelling produces a hypothesis $h \in F_n^r$ that has error at most ε.

Proof: Since we defined the pruning error over the whole instance space, the pruning error for the consistent decision tree can be considered as a classification noise over the random sample without violating the Angluin and Laird framework. The number of rules N in a decision tree of rank at most r is bounded above by Lemma VI.0. Findmin(S) produces a consistent decision tree and Prune(r,k,Q,S) minimizes the probability of misclassification over the whole instance space with confidence probability 1. Lemma VI.1 holds for any probability distribution D with $\eta_b < 0.5$ and for any algorithm which minimizes the misclassification over the whole instance space with confidence probability $1 - \delta$ which is less than 1. Letting $\eta_b = \mu_{n,r}$, and noting that $\mu_{n,r}$ is less than 0.5 for all $n \geq r > 0$, produces the desired result.

■

Combined with the result of Findmin(S) in Theorem III.6 and our analysis of Prune(r,k,Q,S) in Theorem V.7, Theorem VI.2 shows that the decision tree with rank at most r on n variables can be learned with pruning with accuracy $1-\varepsilon$ and confidence $1-\delta$ in time polynomial in $1/\varepsilon$, $1/\delta$, $1/(1-2\mu_{n,r})$ and n for fixed rank r, allowing one unit of time to draw each random examples. Thus pruning has increased the number of examples we must

obtain, but still retains its polynomial sampling characteristic. If the rank of the induced decision tree, k, can be estimated probabilistically, Theorem VI.2 can be tightened by replacing $\mu_{n,r}$ by $\mu_{k,r}$.

To reduce the number of examples sufficient for PAC-identification in finite domains we may use Laird's [1987] tighter bound for $0 < \varepsilon, \delta < 1/2$.

Lemma VI.3 *(A tighter bound: [Laird, 1987]):*

Assume $0 < \varepsilon, \delta < 1/2$. For any finite set of N rules,
if we draw a sequence of

$$m \geq [1/\{\varepsilon(1-\exp(-(0.5)(1-2\eta_b)^2))\}] \ln(N/\delta).$$

random examples for target concept f from an arbitrary probability distribution D with classification noise η (an upperbound is $\eta_b < 1/2$) and find any hypothesis h that minimizes the probability of misclassification in the examples, then

$$\text{Prob}\{\ d(h,f) \geq \varepsilon\ \} \leq \delta.$$

Following Corollary VI.4 is obtained from Theorem VI.2 and Lemma VI.3.

Corollary VI.4: For any $n \geq r > 0$, any target function $f \in F_n^p$, $p \leq n$, an equally likely probability distribution D on X_n and any $0 < \varepsilon, \delta < 1/2$, given a sample S derived from a sequence of

$$m \geq [1/\{\varepsilon(1-\exp(-(0.5)(1-2\mu_{n,r})^2))\}]\{(en/r)^r \ln(8n)+\ln(1/\delta)\}$$

random examples of f chosen independently according to D, with probability $1-\delta$, Findmin(S) and Prune(r,k,Q,S) using tree labelling produces a hypothesis $h \in F_n^r$ that has error at most ε.

VII. Other Pruning Rules

In Section V, we determined the least upperbound of the pruning error with the assumption of an equally likely distribution and the tree labelling rule. Here we give additional insight into the pruning error by considering some problematic cases.

In Section V, we saw that sample labelling under a non-uniform distrib-

ution could give a pruning error greater than 0.5. Now consider the sample labelling rule under the assumption of an equally likely distribution. Table 4 presents an example which shows the least upperbound of the pruning error may be greater than 0.5. We use the same notation as in Table 2 and 3 of Section V.

Table 4: Sample labelling with a uniform distribution

x	f(x)	D(x)	n(x)	$f_Q(x)$	$f_{P(Q)}(x)$
(1,1,1,1)	1	1/16	1	1	1
(1,1,1,0)	0	1/16	1	0	0
(1,1,0,1)	1	1/16	1	1	0
(1,1,0,0)	0	1/16	2	0	0
(1,0,1,1)	1	1/16	1	1	0
(1,0,1,0)	1	1/16	0	1	0
(1,0,0,1)	1	1/16	1	1	0
(1,0,0,0)	0	1/16	3	0	0
(0,1,1,1)	1	1/16	2	1	0
(0,1,1,0)	1	1/16	0	1	0
(0,1,0,1)	1	1/16	0	1	0
(0,1,0,0)	1	1/16	0	1	0
(0,0,1,1)	1	1/16	1	1	0
(0,0,1,0)	1	1/16	0	1	0
(0,0,0,1)	1	1/16	1	1	0
(0,0,0,0)	0	1/16	6	0	0

In this decision tree, r(Q) = 2 and r(P(Q)) = 1. The pruning error in this case is

Prob { x : $f(x) \neq f_{P(Q)}(x)$ } -

Prob { x : $f(x) \neq f_Q(x)$ }

= 7/16 + 3/16 + 1/16 = 11/16 > 0.5.

We now turn to labelling rules that are not deterministic. In such methods we may use some probabilistic labelling procedure where the pruned node has a probability of the target function f(x) = 0 and f(x) = 1 in proportion to a corresponding sample or instance. As we will see, the pruning error may exceed 0.5 for both sample labelling and tree labelling. Below we define two non-deterministic labeling methods.

Definition VII.0 *(Non-deterministic Labelling):* Let i be an internal node in a decision tree Q, and Q(i) be a subtree of Q such that node i is the root of the tree Q(i).

1. Sample labelling:

Let s(i) be a subset of the sample

S used to construct Q from its root to node i.

Let $s_0(i)$ denote the number of negative examples in s(i) and $s_1(i)$ denote the number of positive examples in s(i). If we prune Q at i, then label i as

$\lceil$ 0 with probability $s_0(i) / (s_0(i) + s_1(i))$ and

$\lfloor$ 1 with probability $s_1(i) / (s_0(i) + s_1(i))$.

2. Tree labelling:

Let $v_{i1},...,v_{ip}$ be the nodes in the path from the root to the parent of node i in Q. And let $a_{i1},...,a_{ip}$ be the labels of each node, where $i_1,...,i_p$ are p distinct indices of {1,...,n}.

Let $l_0(i) = |\{x: f_Q(x)=0\}|$, where $x \in X_n$ such that $a_{i1},...,a_{ip}$ are the same.

Let $l_1(i) = |\{x: f_Q(x)=1\}|$, where $x \in X_n$ such that $a_{i1},...,a_{ip}$ are the same. If we prune Q at i, then label i as

$\lceil$ 0 with probability $l_0(i) / (l_0(i) + l_1(i))$ and

$\lfloor$ 1 with probability $l_1(i) / (l_0(i) + l_1(i))$.

By using the case shown in Table 4, we show that the pruning error may exceed 0.5 with both non-determ inistic labelling methods. If we use non-deterministic sample labelling, then the label of the pruned node may have a 0 or 1 label with the corresponding probabilities defined in Definition VII.0. Since $s_0()$ are positive for all pruned nodes of P(Q) in the case shown in Table 4, all pruned nodes may have 0 labels. In that case the pruning error is

$$= 0.5(0.875-0) + 0.25(0.75-0) + 0.125(0.5-0)$$
$$= 0.6875 > 0.5.$$

Similarly, since $l_0()$ are positive for all pruned nodes of P(Q), all pruned nodes may have 0 labels. In that case the pruning error for tree labelling is equal to that for sample labelling calculated above. So, even for an equally likely distribution D, the least upperbound of the pruning error may exceed 0.5 under non-deterministic labelling.

However, if we use the non-determi nistic tree labelling rule, the upperbo- und of the average pruning error is

less than or equal to 0.5 under a uniform distribution. The following lemma shows this.

Lemma VII.1: When D is a uniform distribution and Prune() uses the non-deterministic tree labelling rule, the upperbound of the average pruning error, $m_{k,r}$, is less than or equal to 0.5 for all $k>r\geq 0$.

Proof: Let $Q \in W_n^k$. Suppose some internal node "i" in Q is at level m ($m<n$). Further suppose that we prune Q to P(Q) at node "i" leaving "i" a leaf node in P(Q). Let p("i") be the probability of instances covered by the leaves of the subtree Q(i) in Q. Under a uniform distribution, $p("i") = 2^{n-m}/2^n = (0.5)^m$. Let e("i") be the probability of incorrect classifications of the instances covered by the leaves of subtree Q(i) over the whole instance space.

By the definition of $l_0(i)$ and $l_1(i)$ in Definition IV.1, $l_0(i)+l_1(i) = 2^{n-m}$. Suppose $\alpha_0(i)$ is the number of incorrect classifications of $l_0(i)$ and $\alpha_1(i)$ is the number of incorrect classifications of $l_1(i)$. Then clearly

$$e("i") = (0.5)^m (\alpha_0(i) + \alpha_1(i)) / (l_0(i) + l_1(i))$$

where $0 \leq \alpha_0(i) \leq l_0(i)$ and $0 \leq \alpha_1(i) \leq l_1(i)$.

If we prune Q at node "i", then in P(Q) the label of the leaf "i" is

$$\begin{cases} 0 & \text{with probability } l_0(i) / (l_0(i) + l_1(i)) \text{ and} \\ 1 & \text{with probability } l_1(i) / (l_0(i) + l_1(i)). \end{cases}$$

Below we suppress (i) for clearer reading. If the label of "i" is 0, then

$$e("i") = (0.5)^m (1 - (l_0 - \alpha_0 + \alpha_1) /(l_0 + l_1))$$

since the true probability of a 0 label is $(l_0 - \alpha_0 + \alpha_1) / (l_0 + l_1)$.

By the definition of the pruning error,

$$\begin{aligned} e(Q,P(Q)) &= e(P(Q)) - e(Q) \\ &= (0.5)^m[((l_1 + \alpha_0 - \alpha_1) - (\alpha_0 + \alpha_1)) / (l_0 + l_1)] \\ &= (0.5)^m(l_1 - 2\alpha_1) / (l_0 + l_1). \end{aligned}$$

Similarly, if the label of node "i" is 1, then

$$e(Q,P(Q)) = (0.5)^m (l_0 - 2\alpha_0) / (l_0 + l_1).$$

So, the average pruning error is

$= (l_0 / (l_0 + l_1)) * (0.5)^m (l_1 - 2\alpha_1)$
$/ (l_0 + l_1) + (l_1 / (l_0 + l_1))$
$* (0.5)^m (l_0 - 2\alpha_0) / (l_0 + l_1)$
$= (0.5)^m (2l_0l_1 - 2l_0\alpha_1 - 2l_1\alpha_0)$
$/ (l_0 + l_1)^2$
$\leq (0.5)^m (2l_0l_1) / (l_0 + l_1)2$
$\leq (1/2)(0.5)^m$

since $4l_0l_1 \leq (l_0 + l_1)^2$ and $\alpha_0, \alpha_1 \geq 0$.

Since the above bound holds for all pruned nodes, the upperbound of the average pruning error, $m_{k,r}$, is less than or equal to 0.5.

■

So, the non-deterministic tree labelling rule can hedge the risks of biased non-random selections of examples by guaranteeing that the average pruning error does not exceed 0.5 regardless of the sampling distribution.

VIII. Summary and Conclusions

Empirical results have shown that pruning can improve the accuracy of an induced decision tree. It also leads to more concise rules. Here we provide a pruning algorithm based on the rank of a decision tree. A bound on the error due to pruning by the rank of a decision tree is determined under the assumptions of an equally likely distribution of the instance space and a deterministic tree labelling rule. This bound is then used with recent results in learning theory to determine a sample size sufficient for PAC identification of decision trees with pruning. We also discuss other pruning rules and their effects on the error due to pruning. With non-deterministic tree labelling rule we show that the upperbound of the average pruning error is less than or equal to 0.5 under an equally likely distribution.

Future work will be needed to determine the pruning error under more general assumptions on the distribution over the instance space. Also, Theorem VI.2 can be tightened if an a prior estimate, k, of the rank of the induced decision tree can be determined. If so, $\mu_{n,r}$ can be replaced by $\mu_{k,r}$.

Here we take the rank of a tree as a conciseness measure of a decision tree. Future work will be needed to assess the effect of pruning under other conciseness criteria.

REFERENCES

Angluin, D. and Laird, P. "Learning From Noisy Examples",*Machine Learning*, 2, 1983, pp. 343-370.

Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M. "Occam's Razor", *Information Processing Letters*, 24, 1987, pp.377-380.

Braun, H. and Chandler, J.S. "Predicting Stock Market Behavior through Rule Induction: An Application of the Learning-from-Example Approach", *Decision Sciences*, 18, 1987, pp.415-429.

Breiman, L., Freidman, J., Olshen, R. and Stone, C. *Classification and Regresson Trees*, Wadsworth International, California, 1984.

Carter, C. and Catlett, J. "Assessing Credit Card Applications using Machine Learning", *IEEE Expert*, Fall 1987, pp. 71-79.

Chan, P.K. "Inductive Learning with BCT", *Proceedings of the 6th international Workshop on Machine Learning* (pp. 104-108), Morgan Kaufmann, San Mateo, CA, 1989.

Cheng, J., Fayyad, U.M., Irani, K.B. and Qian, Z. "Improved Decision Trees A Generalized Version of ID3", *Proceedings of the 5th International Conference on Machine Learning* (pp. 100-106), Morgan Kaufmann, San Mateo, CA, 1988.

Ehrenfeucht, A. and Haussler, D. "Learning Decision Trees From Random Examples", *Proceedings of the 1988 workshop on Computational Learning Theory* (pp. 182-194), Morgan Kaufmann, San Mateo, CA, 1988.

Fisher, D. H. and Schlimmer, J. C. "Concept Simplification and Prediction Accuracy", *Proceedings of the 5th International Conference on Machine Learning* (pp. 22-28), Morgan Kaufm-

ann, San Mateo, CA, 1988.

Haussler, D. "Quantifying Inductive Bias AI Learning Algorithms and Valiant's Learning Framework," *Artificial intelligence*, 36, 1988, pp.177-221.

Laird, P. *Learning from good data and bad.* Doctoral Dissertation, Department of Computer Science, Yale University, New Haven, CT, 1987.

Messier, W.F. and Hansen, J.V. "Inducing rules for Expert Systems Development", *Management Science*, 34(12), 1988, pp.1403-1415.

Michalski, R.S. and Chilausky, C. "Learning by being told and learning from examples: An Experimental comparision of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis", *International Journal of Policy Analysis and Information Systems*, 4, 1980 pp.125-161.

Mingers, J. "An Empirical Comparison of Selection Measures for Decision Tree Induction", *Machine Learning*, 3, 1989a, pp.319-342.

Mingers, J. "An Empirical Comparison of Pruning Methods for Decision Tree Induction", *Machine Learning*, 4, 1989b, pp.227-243.

Quinlan, R. "Discovering Rules from large collection of examples: A case study" In D. Michie(Ed.), *Expert systems in the microelectronic age.* Edinburgh University Press, Edinburgh, 1979.

Quinlan, R. The effect of Noise in Concept Learning, In R.S. Michalski, J. Carbonell, T. Mitchell(Eds.), Machine Learning: An Artificial Intelligence Approach (Vol. II), Morgan Kaufmann, San Mateo, CA, 1986a.

Quinlan, R. "Induction of Decision Trees," *Machine Learning*, 1, 1986b, pp.86-106.

Quinlan, R. "Simplifying Decision Trees", *International Journal of Man-Machine Studies*, 27, pp.221-234.

Quinlan, R. (1987b), "Generating Production Rules from Decision Trees", *Proceedings of the 10th International joint Conference on Artificial Intelligence* (pp. 304-307), Morgan Kaufmann, Los Altos, CA, 1987b.

Quinlan, R. and Rivest, R.L. "Inferring Decision Trees Using the Minimum Description Length Principle", *Information and Computation*, 80, 1989,

pp.227-248.

Spangler, S., Fayyad, U.M. and Uthurusamy, R. "Inductionof Decision Trees from Inconclusive Data", *Proceedings of the 6th International Conference on Machine Learning*(pp.146-150), Morgan Kaufmann, San Mateo, CA, 1989.-

Utgoff, P. "Incremental Induction of Decision Trees", *Machine Learning*, 4, 1989, pp. 161-186.

Valiant, L.G. "A Theory of the Learnable", Communications of the ACM., 27(11), 1984, pp.1134-1142.

◇ 저자소개 ◇

저자 김현수는 현재 한국정보문화센타에서 정책연구부장으로 재직하고 있다. 서울 대학교 원자핵 공학과를 졸업하고, 한국과학기술원 경영과학과에서 석사, 미국 University of Florida 에서 경영학 박사 학위를 취득하였으며 주요 관심분야는 시스템 분석 및 설계, 전문가 시스템, 데이타 통신, 정보화 정책 분야 등이다.