What Should Be Minimized in a Decision Tree?

Usama M. Fayyad and **Keki B. Irani**
Artificial Intelligence Laboratory
Electrical Engineering & Computer Science Department
The University of Michigan
Ann Arbor, MI 48109-2122
umf@zip.eecs.umich.edu Irani@caen.engin.umich.edu

Abstract

In this paper, we address the issue of evaluating decision trees generated from training examples by a learning algorithm. We give a set of performance measures and show how some of them relate to others. We derive results suggesting that the number of leaves in a decision tree is the important measure to minimize. Minimizing this measure will, in a probabilistic sense, improve performance along the other measures. Notably it is expected to produce trees whose error rates are less likely to exceed some acceptable limit. The motivation for deriving such results is two-fold:

1. To better understand what constitutes a good measure of performance, and
2. To provide guidance when deciding which aspects of a decision tree generation algorithm should be changed in order to improve the quality of the decision trees it generates.

The results presented in this paper can be used as a basis for a methodology for formally proving that one decision tree generation algorithm is better than another. This would provide a more satisfactory alternative to the current empirical evaluation method for comparing algorithms.

1. Introduction

Knowledge Acquisition is one of the important areas of application for machine learning techniques [Quin86]. With the advent of rule-based and expert systems, and their widespread use, came the reality of the knowledge acquisition bottleneck [Feig81]. It still remains to be seen whether machine learning will alleviate the knowledge acquisition bottleneck by endowing machines with the autonomous ability to construct their own knowledge bases. However, some steady advances in this direction have taken place. For example, Michalski conducted experiments in Soybean disease diagnosis that showed that his concept learning program attained a higher level of classification accuracy than a human knowledge acquisition approach conducted as part of the same experiment[Mich78]. Mozetic was able to reduce the size of a ECG database by 97% without loss of information using the same concept learning technique[Moze86]. Quinlan's success with a large database of thyroid problem data is another of the many examples of machine knowledge acquisition[Quin86, Mich79]. For applications in the domain of semiconductor manufacturing see [Iran90].

An important problem associated with applying induction programs is the measurement of performance and the establishment of improvement. So far, different methods for rule induction have been compared against each other empirically. A domain is chosen, and examples are collected. Programs are then used to induce classification rules from a training subset of examples. Performance is measured by using the induced classification rules on a separate set of test examples. The problem with this method of evaluation is that it is time consuming and often inconclusive. Some programs may do better on one data set or domain, and worse on another. In this paper, we attempt to provide a more theoretical basis for establishing improvement for decision tree generation. By examining the different performance measures, we show that some of them that are data dependent (require a test set of examples) can be probabilistically related to a measure that does not require test sets, namely the number of leaves in the induced tree. This may then serve as a basis for provably establishing that one method for inducing decision trees is better than another by proving that one algorithm always produces a tree with a smaller number of leaves than another algorithm, given the same training data. In addition to giving a possible tool for proving that one program is better than another, the results provide some guidance to ways of improving decision tree learning algorithms.

2. Inducing Decision Trees From Examples

Programs that learn from examples are presented with examples of different concepts (classes). The program is provided with the class to which each example belongs. The goal of the program is to construct an effective method (classifier, or set of rules) for classifying examples in their proper classes. The important performance criteria regarding the produced classifier are: size/complexity and error rate. Since programs that learn from examples perform induction, it is not reasonable to require that they produce classifiers that are completely correct. The error rate of a classifier (see definition 1) is strongly dependent on the choice of training set presented to the learning algorithm. On the other hand, finding a classifier with minimal size/complexity for a given data set can easily be shown to be an NP-hard problem[1]. Most systems in the literature, therefore, resort to heuristic approaches that attempt to produce compact classifiers. A particularly efficient and popular approach to inducing classifiers is via the generation of decision trees[Brei84, Quin86, Quin87, Chen88].

In this paper, we show that a compact classifier is indeed a good target to strive for. We demonstrate that, in a certain probabilistic sense, the compactness of a classifier is related to its expected error rate. Results to this effect for general learning algorithms have appeared before in [Blum87]. In this paper we show how those results apply to decision trees in particular, and how they can be used in deciding which performance measure is most critical. Consequently the results point out what to focus on when attempting to modify and "improve" a particular algorithm for inducing decision trees.

First we briefly describe what a decision tree is. For a more detailed discussion, see [Quin86]. A training set consists of examples expressed as attribute-value pairs. Each example belongs to a single class. A decision tree is a tree in which nodes represent subsets of examples and the outgoing branches from a single node constitute conditions which involve a single attribute. The conditions are *mutually exclusive and collectively exhaustive*, thus they induce a partition on the set of examples at the node. The subset of examples represented by any node in the decision tree consists of all examples in the training set that satisfy the conjunction of the conditions appearing on the branches on the path from the root to that node. Typically, the leaf nodes have examples that belong to exactly one class. That class is referred to as the outcome predicted by the leaf. In some cases it may not be possible or desirable to produce a tree that classifies the training set perfectly (i.e. one outcome per leaf, c.f. [Span89]). However, we restrict our attention to the case where perfect classification is possible.

Assumptions

In this paper, it is assumed that the learning algorithm generates a decision tree that classifies all examples in the *training set* correctly. Another assumption we make is that the sets of training and test examples are *noise free* and are *not ambiguous*[2]. Furthermore, we assume that the examples in the training and test sets are drawn independently and at random according to some fixed (unknown) probability distribution. Finally, the training examples are expressed in terms of attribute-value pairs (see section 4) and the conditions on the outgoing branches of a node are mutually exclusive and collectively exhaustive tests on the values of a single attribute. These assumptions admit most decision tree algorithms in the literature such as CART[Brei84], ID3[Quin86], GID3[Chen88], and others.

3. Performance Evaluation Criteria

There are several dimensions along which the complexity and correctness of a decision tree can be measured or estimated. Quinlan [Quin87] used the number of nodes in the tree as a measure of its optimality. He also used the percentage error rate when the tree is used to classify examples not in the training set. A decision tree may also be translated into a set of rules that perform an equivalent classification. Some performance measures may be defined on this set of classification rules as well. Below, we list measures of performance that we used in a previous paper[Chen88]. The (↑) and (↓) symbols indicate that it is desirable (intuitively) for the corresponding measure to be maximized or minimized respectively.

M1: Percentage errors on classifying unseen examples (↓).

M2: Number of rules in the rule-base (leaves in tree) (↓).

M3: Number of nodes in the tree (↓).

M4: Total number of preconditions in the rule-base (↓). This measures the generality of the entire set of rules. It is a more accurate measure of generality than the average number of preconditions per rule (M6), since the latter is not very indicative if the number of rules is relatively large when compared against the number of training examples.

M5: Average example support per rule (per leaf) (↑). This is the average number of examples in the *test set* on which a rule is applicable and correctly predicts the class. It is a measure of the applicability (generality or utility) of the produced rules. [3]

M6: Average number of tests per example in the test set (↓). This is a data-dependent measure of the efficiency of the generated classification tree (the weighted average depth of the tree).

Note that measures M1, M4, M5, and M6 are essentially estimated by using the set of rules to classify several test sets of examples whose classification is known. Performance measure M4 can be directly related to measure M2. The relation between M3 and M2 is typically direct but can only be quantified when the algorithm is fixed (i.e. it depends on the form of trees generated by the algorithm).

The percentage of errors (measure M1) serves as an estimate of the *true error rate* of the set of rules (see definition 1). It is considered by most researchers in the

[1]It is essentially the multiple-valued logic minimization problem, also see [Hyaf76].

[2]A data set is *ambiguous* when there are at least two examples that agree on the values of all the attributes but belong to different classes.

[3]This measure should be normalized by the total number of examples in the test set for which a prediction, correct or erroneous, was made.

field to be the key measure. Afterall, a minimal set of rules which has a high error rate is not beneficial. On the other hand, a lower error rate is desirable even at the expense of a slightly less efficient or a larger set of rules. However, obtaining a good estimate of the error rate of a decision tree is not easy and it necessitates the use of separate large test sets of examples. A major theme of this paper is to show that by minimizing the number of leaves (measure M2), one may, *in a probabilistic sense*, expect a reduction in error rate to result. Measure M5 will also be shown to be directly related to measures M1 and M2. Measure M6 is not as easy to analyze. We shall delay discussing it until section 5.

The above discussion indicates that it is important to try to minimize the number of leaves in the tree. By reducing the number of leaves we should in general expect an improvement in performance along the other measures listed above. We justify this claim in the next section. This claim is not intuitively obvious. Generally speaking, it is not clear which performance measure is most critical. Although it is highly desirable to have a low error rate, the error rate of a generated tree is not something that can be controlled directly by the learning algorithm. On the other hand, the number of leaves or the depth of the generated tree may be more easily changed by modifying the algorithm. In this sense, it is useful to derive relations between the error rate of a tree and more accessible measures such as the number of leaves in the tree.

4. The Fewer the Leaves, the Better the Tree

First we establish the easy result that shows that measure M5 is directly related to measures M2 and M1. Let n be the number of leaves in a decision tree. Given a test set of t_s examples, measure M5 is defined as[4]
Average example support per rule =

$$\frac{\text{number of test examples correctly classified}}{\text{number of leaves}}$$

In general, given an error rate ϵ (measure M1), we expect $(1-\epsilon)\cdot t_s$ test examples to be classified correctly by the tree. Thus

$$\text{Average example support per rule} = \frac{(1-\epsilon)\cdot t_s}{n}$$

Thus reducing the number of leaves n improves performance along measure M5. Furthermore, reducing the error rate ϵ will also improve the average example support per rule. An objection that may arise here is that decreasing the number of leaves may result in a tree which has a higher error rate. However, our result below shows that, probabilistically, reducing the number of leaves is expected to result in lowering the error rate. This correlation serves to strengthen our statement regarding the relation between the average support per rule (M5) and the number of leaves (M2).

[4]As mentioned before, we are ignoring the normalization factor since it does not affect the analysis.

We now proceed to establish the result regarding the error rate and its relation to the number of leaves in a tree. We first need to establish a few definitions and facts.

Let E be a training set of examples expressed in terms of k attributes from the set $A = \{A_1, \ldots, A_k\}$, and j classes from the set $C = \{C_1, \ldots, C_j\}$. Thus each example $e \in E$ is a $k+1$-tuple

$$e = \langle V_1, \ldots, V_k; C_j \rangle,$$

where $V_i \in \text{Range}(A_i)$ is a value in the range of the attribute $A_i \in A$ and $C_j \in C$. Let K_v be the total number of attribute-value pairs in the training set[5], that is $K_v = \sum_{i=1}^{k} |\text{Range}(A_i)|$.

From this point on, the reader should assume that there is a fixed (unknown) probability distribution according to which training and test examples are draw. Any reference to error rates is implicitly to be interpreted as *error rate with respect to the fixed probability distribution.*

Definition 1: A decision tree T has *error rate* ϵ, $0 \le \epsilon \le 1$, if the probability that T will misclassify a randomly drawn test example (according to our fixed distribution) is ϵ. Equivalently, if T classifies m randomly drawn examples then for very large m, $(m \cdot \epsilon)$ examples are expected to be misclassified by T.

Lemma 1 *For every decision tree T with n leaves, there exists a binary decision tree T' with n leaves that is logically equivalent to it.*

The proof is given by an algorithm that transforms an r-ary decision tree to an equivalent binary one with the same number of leaves. We do not include the proof in this paper. The only requirement is that the conditions on the outgoing branches of a node be mutually exclusive and collectively exhaustive. Clearly two decision trees that are logically equivalent have exactly the same error rate when used to classify a set of examples. This lemma allows us to concentrate our analysis on binary trees only. This simplifies the analysis but does not change the final results.

Definition 2: Let $DT(n)$ denote the set of logically non-equivalent binary decision trees having n leaves.

Two binary decision trees are *logically equivalent* if the two DNF expressions obtained by taking the disjunction of the conditions on the leaves of each tree are equivalent.

Proposition 1 *There are at most $DT_u(n)$ and at least $DT_l(n)$ binary decision trees in $DT(n)$ (logically non-equivalent binary decision trees with n leaves), where*

$$\begin{aligned} DT_u(n) &= K_v^{(n-1)}\frac{1}{n}\binom{2(n-1)}{n-1} \\ DT_l(n) &= \binom{K_v}{n-1}\frac{(n-1)!}{n}\binom{2(n-1)}{n-1} \end{aligned}$$

[5]This holds for discrete attributes. For a continuous attribute A_i, $|\text{Range}(A_i)|$ is at most $|E| - 1$.

The proof consists of counting binary tree topologies and then counting the possible labellings of the internal nodes with the interpretation that the left branch out of an internal node is labelled TRUE and the right branch is labelled FALSE. Note that the upper and lower bounds are fairly tight. For $K_v \geq n > 2$ they differ by at most $(\frac{K_v}{n})!$.

Definition 3: Let $B(n,\epsilon) \subseteq DT(n)$ denote the set of binary trees with n leaves that have an error rate greater than ϵ, $0 < \epsilon < 1$.

We are now ready to derive our main result. The result follows almost directly from a theorem that relates the number of examples needed to probabilistically guarantee a certain error rate. The theorem is by Blumer, Ehrenfeucht, Haussler, and Warmuth and appears in [Blum87]. For completeness, we include the modified proof for trees.

Theorem 1 *Let T be a binary tree having n leaves that classifies a set of m randomly chosen training examples correctly. Then $|B(n,\epsilon)| \cdot (1-\epsilon)^m$ is an upper bound on the probability that T has an error rate greater than ϵ, $0 < \epsilon < 1$.*

Proof: By proposition 1 there are $|DT(n)|$ possible binary trees with exactly n leaves. Let $B(n,\epsilon) \subseteq DT(n)$, where

$$B(n,\epsilon) = \{T \in DT(n) \mid T \text{ has error rate } > \epsilon\}$$

is the set defined in definition 3. We want to bound the probability that our tree T is a member of $B(n,\epsilon)$. For any $T \in B(n,\epsilon)$, and any randomly chosen example $e \in E$,
Prob($\{T$ misclassifies $e\}) > \epsilon$.
Hence, Prob($\{T$ correctly classifies $e\}) < (1-\epsilon)$.
Assuming the m training examples were chosen independently at random according to some fixed probability distribution,

Prob($\{T$ is consistent with the training set$\}) < (1-\epsilon)^m$

We want to bound the probability of *any* tree in $B(n,\epsilon)$ being consistent with m examples, namely

$$\text{Prob}\left(\bigcup_{T \in B(n,\epsilon)} \{T \text{ is consistent with } m \text{ examples}\} \right)$$

By the subadditivity property, this probability is

$$\begin{aligned} &< |B(n,\epsilon)| \cdot \text{Prob}\left(\{T_n \text{is consistent with } m \text{ examples}\}\right) \\ &= |B(n,\epsilon)| \cdot (1-\epsilon)^m \end{aligned}$$

□

Note that $|B(n,\epsilon)|$ is not easy to quantify. To get a looser quantitative bound, one may use the fact that $B(n,\epsilon) \leq |DT(n)|$ by definition, and proposition 1 to show that the probability that the obtained tree has error greater that ϵ is

$$\begin{aligned} &< |DT(n)| \cdot (1-\epsilon)^m \\ &< K_v^{(n-1)} \frac{1}{n} \binom{2(n-1)}{n-1} \cdot (1-\epsilon)^m \end{aligned}$$

However, we do not want to loosen our bound unnecessarily for reasons that should shortly become apparent.

Lemma 2 *For any fixed ϵ, $0 < \epsilon < 1$, and any $n_2 > n_1 \geq 2$ the following property holds:*

$$\frac{|B(n_2,\epsilon)|}{|B(n_1,\epsilon)|} > 2^{(n_2-n_1)}$$

The proof is by induction on the fact that for any $n \geq 2$

$$\frac{|B(n+1,\epsilon)|}{|B(n,\epsilon)|} > 2$$

The proof of this fact is omitted. □

Corollary 1 *Let T_1 and T_2 be two decision trees consistent with a fixed training set of m examples. Let n_1 and n_2 be the number of leaves in T_1 and T_2 respectively. For a fixed ϵ, $0 < \epsilon < 1$, let b_1 and b_2 be the bounds derived in Theorem 1 for T_1 and T_2 respectively.*
If $n_2 > n_1 \geq 2$ then $b_1 < b_2$. Furthermore

$$\frac{b_2}{b_1} > 2^{(n_2-n_1)}.$$

Proof: $n_1 < n_2$, $0 < (1-\epsilon) < 1$, and Theorem 1 give $b_1 < b_2$. b_2 being larger than b_1 exponentially in the difference $(n_2 - n_1)$ follows from Theorem 1 and Lemma 2 regarding growth of $|B(n,\epsilon)|$ with n. □

Thus for a fixed training set, given two decision trees T_1 and T_2 with n_1 and n_2 leaves respectively, let $P_1 = \text{Prob}(\{T_1 \text{has error rate } > \epsilon\})$ and $P_2 = \text{Prob}(\{T_2 \text{has error rate} > \epsilon\})$. Note that if $n_1 < n_2$ it follows from Theorem 1: $P_1 < b_1$ and $P_2 < b_2$. Corollary 1 states that b_1 is smaller than b_2 by a factor of $2^{(n_2-n_1)}$. However, Corollary 1 *does not* imply that $P_1 < P_2$. Proving this would be desirable but is not possible because the trees T_1 and T_2 were derived by an induction process that examined the same finite subset of the set of all possible examples. The corollary *does* state that the *upper bound* on the probability that T_1 has an error rate that exceeds ϵ is always lower than the corresponding upper bound for T_2. Note that the upper bounds grow exponentially tighter with the number of training examples available to the learning program. Finally, even with very tight upper bounds (i.e. very large training sets) b_1 is always smaller than b_2 by a factor that grows exponentially in the difference $(n_2 - n_1)$. Thus improvement in the upper bound on the probability of error is very sensitive to a decrease in the number of leaves of the generated tree for a fixed training set.

We shall denote the probability that a tree T has error greater than ϵ by $P(T,\epsilon)$, i.e.

$$P(T,\epsilon) = \text{Prob}\left(\{T \text{ has error rate} > \epsilon\}\right).$$

Definition 4: We say that a tree T_1 is *likely to have a lower error rate* than a tree T_2 if, for a fixed ϵ, $0 < \epsilon < 1$,

$$\text{Prob}\left(\{P(T_1,\epsilon) < P(T_2,\epsilon)\}\right) > \frac{1}{2}$$

i.e. when it is more likely that $P(T_1,\epsilon) < P(T_2,\epsilon)$ than otherwise.

Given two decision trees, we should prefer the one that is *likely to have a lower error rate* than the other. Assume that, for a fixed training set of m examples, we are given two decision trees T_1 and $T2$ having n_1 and n_2 leaves respectively, with $n_2 > n_1$. If we have no special knowledge of the data or the trees, we only have one more piece of information, namely that for any fixed ϵ, corollary 1 states that the bound b_1 on $P(T_1, \epsilon)$ is exponentially smaller than the corresponding bound $b2$ for $P(T_1, \epsilon)$. How may we use this piece of information? Is there a formally justifiable strategy for preferring T_1 over T_2?

Having no further knowledge, it seems reasonable to assume that $P(T_1, \epsilon)$ could be any value less than b_1. i.e. that $P(T_1, \epsilon)$ is uniformly distributed over the range $[0, b_1)$. Assuming the same for T_2, the following corollary will justify our strategy for preferring T_1 over T_2.

Corollary 2 *Given two decision trees T_1 and T_2 having n_1 and n_2 leaves respectively, if $n_1 < n_2$ then assuming that $P(T_1, \epsilon)$ and $P(T_2, \epsilon)$ are uniformly distributed below there respective bounds b_1 and b_2, then T_1 is likely to have a lower error rate than T_2.*

Proof: From Theorem 1 and corollary 1 we get that $P(T_1, \epsilon) < b_1$, $P(T_2, \epsilon) < b_2$, and that $b_1 < b_2$ by a factor of $2^{(n_2 - n_1)}$.
Now if $P(T_1, \epsilon)$ and $P(T_2, \epsilon)$ are uniformly distributed over the ranges $[0, b_1)$ and $[0, b_2)$ respectively, we can show that

$$\text{Prob}\left(\{P(T_1, \epsilon) < P(T_2, \epsilon)\}\right) > \left[1 - \left(\frac{1}{2} \cdot \frac{1}{2^{(n2-n1)}}\right)\right]$$

which is always greater than 0.75. □

Corollary 2 justifies the strategy that prescribes preferring the tree with the smaller number of leaves. Note that Corollary 2 *does not* state that a tree with the minimal number of leaves is the best tree. The key condition is that the tree must be consistent with the training examples. For example, the tree consisting of a single node is not consistent with the training set (unless all training examples are of one class, in which case it would be the best tree.) Furthermore, the corollary does not state that $P(T_1, \epsilon) < P(T_1, \epsilon)$, it just states that, under the uniform distribution assumption, the event that $\{P(T_1, \epsilon) < P(T_1, \epsilon)\}$ is a more likely event than its complement: $\{P(T_2, \epsilon) < P(T_1, \epsilon)\}$.

The above strategy is motivated by the assumption that we (i.e. the users/designers of the learning algorithm) have no special knowledge of the domain. Since the true error rate is not easy to get, we would like to at least have a certain confidence that a given tree's error rate is "acceptable", namely it is less than ϵ. We thus "prefer" a tree for which a tighter upper bound on the probability of error being "unacceptable" can be established.

5. Discussion

We have shown that by concentrating on improving a single performance measure, namely the number of leaves in the decision tree, we can achieve, probabilistically, improvement along the other performance measures listed in section 3. The usefulness of this result is in pointing out an aspect of decision trees where further research should be emphasized. We know that the problem of finding optimal decision trees (with respect to almost any performance measure) is NP-hard. Thus it is very likely that there exists no efficient algorithm for producing optimal trees. We therefore have to be satisfied with heuristic algorithms that generate suboptimal trees. With the growing usage of these algorithms, we become more familiar with their problems and limitations, and the need arises for improving them. The result given in this paper suggests that if the goal is to improve a certain heuristic algorithm for generating decision trees, then a good approach to take is to try to modify it in a way that causes it to generate trees that have *fewer leaves*. It does not matter how the tree with fewer leaves is generated since the results of this paper predict that the tree is expected to perform better based solely on the fact that it has fewer leaves[6].

There remains two issues that need to be addressed. The first is that the results are probabilistic, and they cannot be absolute. This is due to the fact that the quality of the decision tree relies heavily on the choice of training set. Thus by choosing pathological training sets, one could force a given algorithm to generate trees that are arbitrarily "bad". An absolute statement regarding improvement in performance would therefore necessarily be false. The second is that the bounds used are worst case bounds and may be too restrictive. However, this does not mean they are not useful. Corollary 2 shows that it is indeed a "safer" strategy to prefer the tree with the smaller number of leaves. At this time, we cannot state whether an average-case analysis would necessarily lead to a result consistent with the results presented.

Finally, our analysis did not take into account performance measure M6. The relation between the number of leaves and the average depth of the decision tree (weighted by relative frequencies of examples in test set) is not clear. The latter would favour a tree with a very large branching factor. One extreme of this is if each example had a unique id number, and if the id number were used as an attribute to create a branch for each id number. This creates a tree with depth 1 and a leaf node for each example in the training set. However, this tree will fail to classify any example that did not appear in the training set. On the other hand, measure M6 is a good measure of the speed of the decision tree in classifying an example—the average number of tests required before an example can be classified. If the tests (attributes) are not very costly, then measure M6 can be ignored. In this case a deeper tree means that the rules have more preconditions, which means more information is encoded in each rule, and thus, at least intuitively, one would expect such rules to be better predictors. Thus it is not at all

[6]As long as the tree is consistent with the training set.

clear, at least intuitively, whether this measure should be minimized.

6. Conclusions

The main motivation for this work is to derive results that give us a handle on two problems relating to decision tree generation:

1. If a particular algorithm for inducing decision trees is to be "improved", then what should be changed in it? Out of the various performance measures available, which is most critical?
2. Suppose one gives a new algorithm for generating decision trees, then how can one go about establishing that it is indeed an improvement?

To date, the answer to the second problem has been: Compare the performance of the new algorithm with that of the old algorithm by running both on many data sets. This is a slow process that does not necessarily produce conclusive results. On the other hand, suppose one were able to prove that given a data set, Algorithm A will always (or "most of the time") generate a tree that has fewer leaves than the tree generated by Algorithm B. Then the results of this paper can be used to claim that Algorithm A is "better" than algorithm B. In this case "better" means: without special knowledge of the data, given a training set, it is more likely that the tree generated by Algorithm A will outperform the one generated by Algorithm B on the performance measures considered.

We have presented some results that seem to indicate that the number of leaves in a decision tree is *the* important measure to consider when attempting to show that one tree is better than another if both trees were generated from the same training set. The results hold when no special knowledge of the data is available. All that is required is that the generated tree classify the training set correctly. We have also identified some of the limitations of the derived results. The primary motivation behind this work is to produce the tools needed to make it possible to formally prove improvement in decision tree generation algorithms and thus avoid the less than satisfactory and tedious approach of empirical comparison over large varieties of training and test example sets.

Acknowledgments

The authors wish to thank Kevin Compton for help on counting trees, and to thank Sami Bayyuk, Jie Cheng, and Moncef Maiza for insightful and helpful discussions. This work was supported in part by a DeVlieg Industrial Fellowship and in part by a Hughes Unrestricted Grant.

References

[Blum87] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. "Occam's Razor." *Information Processing Letters 24*. pp. 377-380. North-Holland (1987).

[Brei84] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. *Classification and Regression Trees*. Monterey, CA: Wadsworth & Brooks (1984).

[Chen88] Cheng, J., Fayyad, U.M., Irani, K.B., and Qian, Z. "Improved decision trees: A generalized version of ID3." *Proceedings of the Fifth International Conference on Machine Learning*. pp. 100-108. Ann Arbor, MI (1988).

[Feig81] Feigenbaum, E.A. "Expert systems in the 1980s." In Bond, A. (Ed.), *State of The Art Report on Machine Intelligence*. Maidenhead: Pergamon-Infotech, (1981).

[Hyaf76] Hyafil, L. and Rivest, R. "Constructing optimal binary decision trees is NP-complete." *Information Processing Letters*. vol 5, no. 4. (May 1986).

[Iran90] Irani, K.B., Cheng, J., Fayyad, U.M., and Qian, Z. "Applications of Machine Learning Techniques in Semiconductor Manufacturing." *Proceedings of The S.P.I.E. Conference on Applications of Artificial Intelligence VIII*. Orlando, Fl (1990).

[Mich78] Michalski, R.S. and Larson, J.B. "Selection of most representative training examples and incremental generation of VL1 hypotheses: The underlying and the description of programs ESEL and AQ11." Report No. 867. Computer Science Dept., University of Illinois, Urbana, (1978).

[Mich79] Michie, D. (Ed.) *Expert Systems in the Micro Electronic Age*. Edinburgh: Edinburgh University Press (1979).

[Moze86] Mozetic, I. "Knowledge extraction through learning by examples." In Mitchell, T.M., Carbonell, J.G., and Michalski, R.S. *Machine Learning: A Guide to Current Research*. pp. 227-231. Boston: Kluwer Academic Publishers, (1986).

[Quin86] Quinlan, J.R. "Induction of decision trees." *Machine Learning 1, No. 1*. pp. 81-106. Boston: Kluwer Academic Publishers, (1986).

[Quin87] Quinlan, J.R. "Generating production rules from decision trees". *IJCAI-87*. pp. 304-307. Milan, Italy (1987).

[Span89] Spangler, S., Fayyad, U.M., and Uthurusamy, R. "Induction of decision trees from inconclusive data." *Proceedings of the Sixth International Workshop on Machine Learning*. pp. 146-150. Ithaca, NY (1989).