Machine Learning, 24, 231–242 (1996)

Efficient Incremental Induction of Decision Trees

DIMITRIOS KALLES dk@mp.co.umist.ac.uk
Department of Computation, UMIST, PO Box 88, Manchester, M60 1QD, U.K.

TIM MORRIS dtm@ap.co.umist.ac.uk
Department of Computation, UMIST, PO Box 88, Manchester, M60 1QD, U.K.

Editor: Paul Utgoff

Abstract. This paper proposes a method to improve ID5R, an incremental TDIDT algorithm. The new method evaluates the quality of attributes selected at the nodes of a decision tree and estimates a minimum number of steps for which these attributes are guaranteed such a selection. This results in reducing overheads during incremental learning. The method is supported by theoretical analysis and experimental results.

Keywords: Incremental algorithm, decision tree induction

1. Introduction

A decision tree is a model of the evaluation of a discrete function (Moret, 1982). This model represents a step-by-step computation where, in each step, the value of a variable is determined and according to that value the next action is chosen. Possible actions are the selection of some other variable for evaluation, the output of the value of the function, or the remark that for the particular variable-value combination the function is not defined. In a broad context, a variable may be a combination of other variables rather than a single variable (Breiman et al., 1984; Pagallo, 1989; Brodley and Utgoff, 1995).

This description suggests a way of building such a decision tree: given a set of training instances that represent the values of the function at specific points of the pattern space, the decision tree designer should come up with informative questions about the values of the variables such that each question builds on the results of previous questions and progresses towards the computation of the output of the function as fast as possible. This is the basic concept behind Top-Down Induction of Decision Trees (TDIDT) algorithms.

Building an optimal decision tree is an NP-complete problem (Hyafil and Rivest, 1976; Naumov, 1991), therefore heuristic methods are used. The most notable algorithm is probably ID3 (Quinlan, 1983; Quinlan, 1986), which paved the way for numerous other variations and improvements.

There is a distinct difference between two types of algorithms in this discipline: incremental algorithms are able to build and refine a concept in a step-by-step basis as new training instances become available, whereas non-incremental algorithms work in batch mode (off-line). The relative advantages of incremental techniques are elaborated by Utgoff (1989), who also presents ID5R, an incremental algorithm as an extension to ID3. ID5R serves as the basis for the discussion in this paper.

COBWEB (Fisher, 1987) was one of the early incremental learning systems. ID4 (Schlimmer and Fisher, 1986) and ID5 (Utgoff, 1988) were also proposed before ID5R as extensions to ID3. ID4 may have to reconstruct the tree several times for different training set orderings and ID5 does not guarantee compatibility between the finally delivered tree and the tree produced by ID3 (regardless of which attribute selection criterion the ID3-based algorithm would use). The same can be said for the IDL algorithm (Van de Velde, 1990) that utilizes topological relevance to perform incremental induction and the principles discussed by Cockett and Zhu (1989), where the concept of association reductions is introduced.

The scope of interest of this paper is the study of incremental algorithms that, at each step, guarantee compatibility between the incrementally induced tree and the tree produced by an "off-line" method (for example, ID5R guarantees ID3-compatibility). We discuss a method of improving the learning speed of the ID5R algorithm and at the same time guaranteeing that the tree is constructed by exactly the same sequence of operations that ID5R would perform. The method is analytically evaluated and experimentally validated. The paper concludes by identifying directions for improvements.

2. Overview of the problem

A brief description of the ID5R algorithm is presented below. For details and a complexity analysis of some TDIDT algorithms, the reader is referred to the original paper by Utgoff (1989).

Given a decision tree and a new training pattern to be incorporated into the tree (by incremental learning), one starts by examining the root node of the tree. The attribute, according to which the primary partitioning of the pattern space is made, has a score, which allows it to prevail over competing attributes. However, as the new pattern arrives this score may change along with the scores of competing attributes. This means that at some point, splitting on the original attribute is no longer warranted by the current scores. Should this be the case, the new best attribute is recursively pulled-up to the root of the tree and the original one is demoted. The pull-up is effected by means of *transpositions* (also discussed by Cockett (1987)), which are structural operations that swap attributes in consecutive levels of a decision tree. After the demotion of the original root attribute, the rest of the tree is recursively searched to achieve consistency for all subtrees, taking into account the new pattern and the restructuring. As the pattern is "propagated" towards a leaf node (possibly a new one) the scores of all attributes that are available at each node of the decision path are updated.

The particular drawback of ID5R in its current form (as described) is that, besides the updating of the scores of the competing attributes, the propagation of a pattern along its decision path generates a selection process at each node. Such processes determine whether the current splitting attribute is still the most informative, according to the splitting criterion. In doing this, the algorithm blindly ignores that an attribute may have outperformed its competitors by such a margin that it is impossible for it to turn up as a second best within the next few new patterns. The major topic of the following analysis is to quantify "few". The effects of such a blind reconsideration may be detrimental to the speed of the learning process. This speculation is a motivation to research whether one can guarantee that some

nodes will be *stable* in their selection of attribute tests over a number of steps; after this number of steps one would have to reconsider the situation at that node and possibly pull-up a new attribute without having to worry that this pull-up should have occurred earlier. This is the core of the argument to be pursued.

ID5R has been now superseded by ITI (Utgoff, 1994). ITI is an algorithm that handles numeric attributes and missing values and uses the gain ratio as an attribute selection policy (Quinlan, 1986; Quinlan, 1993). For the sake of simplicity in the following discussion, numeric attributes and missing values are ignored, and the information gain is used. The principle remains the same, however.

3. Description and theoretical analysis of the method

Assume that at a given node two attributes compete and that attribute F_1 prevails over attribute F_2. Assume also that during incremental learning, attribute F_2 is going to be the sole competitor for F_1 (this is not an optimistic bias of the measure of the ability of F_1 to withstand competition). Let E_1 and E_2 be the respective scores of these attributes. The score refers to the measure of goodness of a particular attribute for splitting. It may be a form of the information gain, the Gini index (Breiman et al., 1984), or some other.

The information gain criterion for a two-class problem (positive and negative) will be presented. Extending the analysis to a multi-class problem is straightforward but will be omitted because it is somewhat more complex and does not enhance the understanding of the problem.

Note that for the information gain criterion, if an attribute F_1 prevails over an attribute F_2, the following hold:

$$\begin{aligned} gain(F_1) \geq gain(F_2) &\Leftrightarrow I(p,n) - E(F_1) \geq I(p,n) - E(F_2) \\ &\Leftrightarrow E(F_1) \leq E(F_2) \\ &\Leftrightarrow E_1 \leq E_2 \end{aligned} \tag{1}$$

In the expressions above, p_i is the number of positive instances with value i (n_i refers to the negative instances). $I(p,n)$ is the expected information content of a source which may transmit either 'P' (positive) or 'N' (negative) with probabilities $p/(p+n)$ and $n/(p+n)$ respectively. $E(F)$ is the resulting information content after branching on attribute F (Quinlan, 1986). Hereafter, the score E_i of an attribute F_i will be used to denote the quantity $E(F_i)$.

The purpose of the analysis below is to bound the number of steps before there can be any change in attribute preference between attributes F_1 and F_2 (a new training instance corresponds to one step).

Suppose that a new training instance arrives and that at this given node it is probed (for some attribute) and found to have value j (this instance is also said to belong to bucket j).

The score of the splitting attribute, before the new instance is added, is computed as follows:

$$E = \sum_{i=1}^{m} \frac{p_i + n_i}{p+n} \left(-\frac{p_i}{p_i+n_i} \log \left(\frac{p_i}{p_i+n_i} \right) - \frac{n_i}{p_i+n_i} \log \left(\frac{n_i}{p_i+n_i} \right) \right)$$

$$= \frac{\varepsilon}{p+n} \tag{2}$$

where

$$\varepsilon = -\sum_{i=1}^{m} \left(p_i \log \left(\frac{p_i}{p_i + n_i} \right) + n_i \log \left(\frac{n_i}{p_i + n_i} \right) \right) \tag{3}$$

and, of course $\varepsilon \geq 0$.

In the expressions above, m denotes the number of values a given attribute can have. As a convention of notation, *log* will denote a logarithm of base 2. Note also that the terms *instance* and *pattern* will be used as synonyms.

Without loss of generalization, assume that the new instance is positive. The new score is computed.

$$\begin{aligned} E' &= \frac{-1}{p+n+1} \sum_{\substack{i=1 \\ i \neq j}}^{m} \left(p_i \log \left(\frac{p_i}{p_i + n_i} \right) + n_i \log \left(\frac{n_i}{p_i + n_i} \right) \right) + \\ &\quad \frac{-1}{p+n+1} \left(-(p_j + 1) \log \left(\frac{p_j + 1}{p_j + n_j + 1} \right) - n_j \log \left(\frac{n_j}{p_j + n_j + 1} \right) \right) \\ &= \frac{\varepsilon + A}{p+n+1} \end{aligned} \tag{4}$$

where

$$\begin{aligned} A &= p_j \log \left(\frac{p_j}{p_j + n_j} \right) + n_j \log \left(\frac{n_j}{p_j + n_j} \right) - \\ &\quad (p_j + 1) \log \left(\frac{p_j + 1}{p_j + n_j + 1} \right) - n_j \log \left(\frac{n_j}{p_j + n_j + 1} \right) \end{aligned} \tag{5}$$

By setting $x = p_j$ and $y = n_j$, and by some manipulations we obtain:

$$\begin{aligned} A &= x \log x + (x + y + 1) \log(x + y + 1) - \\ &\quad (x + y) \log(x + y) - (x + 1) \log(x + 1) \end{aligned} \tag{6}$$

The difference in the scores, before and after the consideration of the new training instance, is now

$$\begin{aligned} \Delta E &= E' - E \\ &= \frac{\varepsilon + A}{p+n+1} - \frac{\varepsilon}{p+n} \\ &= \frac{A}{p+n+1} - \frac{\varepsilon}{(p+n)(p+n+1)} \end{aligned} \tag{7}$$

Minimum and maximum values are now required for the quantity above; these will allow the computation of a bound on the rate of convergence of scores between two competing attributes.

Consider first the quantity denoted by A. This is a function of two variables, namely $A(x, y)$ (we define that $0 \times \log 0 = 0$ for convenience in the following computations).

LEMMA 1

$$0 \leq A(x, y)$$

Proof: For a point (x_0, y_0) to be a local minimum, it must satisfy the following condition:

$$\frac{\partial A}{\partial x}(x_0, y_0) = \frac{\partial A}{\partial y}(x_0, y_0) = 0 \tag{8}$$

One can easily verify that this condition cannot be met by any (x_0, y_0) point, so local extrema may only appear as instances of a boundary problem. Under the constraints that $x \in [0, +\infty)$ and $y \in [0, +\infty)$ and the fact that x and y cannot be both 0, it follows that $A_{min} = 0$ (since $A(x, 0) = 0$ and $A(0, y) \geq \log(y + 1)$). ■

LEMMA 2

$$A(x, y) \leq \log(x + y + 1) + \log e$$

Proof: By manipulating the expression for A, we obtain

$$(x + y) \log \left(\frac{x + y + 1}{x + y} \right) \leq \log(x + 1) + x \log \left(\frac{x + 1}{x} \right) + \log e \tag{9}$$

It can be proved easily that the sequence

$$n \log \left(\frac{n + 1}{n} \right)$$

converges to $\log e$ and that its minimum value is 1. We then obtain

$$(x + y) \log \left(\frac{x + y + 1}{x + y} \right) \leq \log e \tag{10}$$

$$x \log \left(\frac{x + 1}{x} \right) \geq 1 \tag{11}$$

By substituting inequalities 10 and 11 into inequality 9, the proof is concluded (for $x = 0$ we work on the original inequality and derive directly the result). ■

LEMMA 3

$$\varepsilon \leq p + n$$

Proof: ε is a sum of m terms, each one of the form

$$-x \log\left(\frac{x}{x+y}\right) - y \log\left(\frac{y}{x+y}\right)$$

Using an argument of extreme values, as in Lemma 1, it follows that $\varepsilon_i \leq p_i + n_i$ and we obtain

$$\varepsilon_i \leq p_i + n_i \Rightarrow \varepsilon \leq \sum_{i=1}^{m} (p_i + n_i) = p + n \tag{12}$$

■

Putting all the above bounds together, the expressions for the extreme values of the score differences follow:

$$\Delta E_{min} \geq \frac{-1}{p+n+1} \tag{13}$$

$$\Delta E_{max} \leq \frac{\log(x+y+1) + \log e}{p+n+1} = \frac{\log(b_j + 1) + \log e}{p+n+1} \tag{14}$$

The term $b_j = p_j + n_j$ in inequality 14 denotes the number of patterns directed to bucket j. To cover the worst case, this will be set to the number of patterns in the bucket that holds most of the patterns during the initial distribution. Note that in such a case the information content decreases. [1]

We can now assume that for every one of the following a instances that arrive at this node, the worst case will occur regarding the initially selected attribute and the best case will occur for competing attributes. In such a case we can obtain an overall worst case scenario for that attribute. For reasons of simplicity, we also assume that $a \leq b_j$ (this, again, is an assumption that biases the final estimate pessimistically, as we explicitly state that we do not expect to postpone consideration for more than a steps).

So, in the case where the competing attribute's score is maximally decremented at each step and the current preferred attribute's score grows at a maximum, the score distance that will be closed between the two attributes after a steps is

$$\Delta E_{max,a} - \Delta E_{min,a} \leq \left(\sum_{i=1}^{a} \frac{\log(b_j + i) + \log e}{p+n+1}\right) + \frac{a}{p+n+1} \tag{15}$$

The numerator of the quantity in the sum simply reflects that the current preferred attribute will accumulate all future a instances into bucket j. If we had tried to derive ΔE_{max} and ΔE_{min} for each one of the a steps, then the denominator would increase (it is easy to carry out the original calculations by assuming a start-up population of $p + n + 1$ instances) and

the bound would be stronger, but a closed formula would be very difficult to derive. If we further expand inequality 15 and note that $a \leq b_j$ we obtain

$$\begin{aligned}\Delta E_{max,a} - \Delta E_{min,a} &\leq \left(\sum_{i=1}^{a} \frac{\log(b_j + b_j) + \log e}{p+n+1}\right) + \frac{a}{p+n+1} \\ &\leq \left(\sum_{i=1}^{a} \frac{\log 2 + \log(b_j) + \log e}{p+n+1}\right) + \frac{a}{p+n+1} \\ &\leq \frac{a(\log(b_j) + 2 + \log e)}{p+n+1} \end{aligned} \tag{16}$$

Using the initial score difference, this is transformed into

$$\begin{aligned} E_2 - E_1 \geq \frac{a(\log(b_j) + 2 + \log e)}{p+n+1} &\Leftrightarrow a \leq \frac{(p+n+1)(E_2 - E_1)}{\log(b_j) + 2 + \log e} \\ &\Leftrightarrow a_{max} = \left\lfloor \frac{(p+n+1)(E_2 - E_1)}{\log(b_j) + 2 + \log e} \right\rfloor \end{aligned} \tag{17}$$

The $(p+n+1)$ factor in the above result can be attributed to the distributive nature of the measure of attribute merit used by the information gain.

The above formula allows room for some insight into the behavior of a decision tree. The $(p+n+1)$ factor in the numerator indicates that nodes that are higher up in the tree (near the root) should find it easier to postpone attribute score evaluations, compared to lower nodes, and that improvement should be easier to observe when large data sets are used. The $\log(b_j)$ factor in the denominator indicates that when attributes with many values are used, they may tend to have a low number of instances in each bucket and thus $\log(b_j)$ decreases. This also results in potentially higher values for a_{max} (denoted simply by a, hereafter).

Table 1 presents the enhanced algorithm (steps to identify leaves and other boundary conditions are omitted, for the sake of conciseness). The basic idea is to accommodate an instance in the decision tree and then consider nodes for evaluation of attribute scores. This is also the basic idea in ITI.

4. Experimental verification

The analysis has been verified by experimenting with the following data sets from the Machine Learning Repository (Murphy and Aha, 1995), available via anonymous *ftp* from the *ics.uci.edu* site, in the directory *pub*:

- **kr-vs-kp**: This data set has 3196 instances with thirty-six attributes and nominal attribute values. There are no missing values. It is a database of chess end games, with a king and a rook on one side and a king and a pawn on the other. The task is to decide if the position is a win for the side of the rook.

- **mushroom**: This data set has 8124 instances with twenty-two attributes and nominal attribute values. There exist some missing values, which are treated here as separate

Table 1. Description of enhanced ID5R.

algorithm *ID5R* (adapted from Utgoff (1989))
Input: a decision tree, a training instance.
Output: a revised decision tree.

```
use the decision tree to classify the new instance and add it to the tree
for each node in the classification path decrease its value of a
for each node affected by the new instance
    if its value of a equals 0
        select the most suitable attribute for that node and re-compute a
        if the selected attribute is not already in that node then
            pull-up the attribute from the leaves towards the node
            establish best attributes for all subtrees of the node
```

algorithm *pull-up* (adapted from Utgoff (1989))
Input: a decision tree, an attribute, a node.
Output: a revised decision tree.

```
recursively pull up the attribute to the root of each immediate subtree of the current node
transpose the decision tree at the current node
set the value of a for all subtree root nodes to 0
```

nominal values. It is a database of mushrooms described in terms of their physical characteristics and classified as poisonous or edible.

It should be noted that the analysis presented above applies to symbolic attributes only and that the possible extensions to explicitly handle numeric attributes and missing values have been left as future work. We expect, however, that such extensions involve mainly implementation issues rather than a fundamental re-thinking of the analysis.

Experimentation consisted of building a decision tree incrementally using ITI (the program, supplied by Paul Utgoff, was modified to use the information gain and incorporate the above presented improvements). The experiments were conducted on a Sparcstation II computer. Each data set was used in ten experiments with a different ordering each time. Although the order in which the training instances are presented has no effect on what tree ITI will build, order does have an effect on the amount of computations required to produce the final tree.

A particular feature of the ITI algorithm is that it generates binary tests. In doing so, it effectively creates competition among attribute-value pairs, rather than among attributes. This is an implementation detail that does not affect the analysis presented above. However, binary attributes need special attention as branching on one value is equivalent to branching on the other. With a naive method of handling them, a binary attribute that would produce

the best attribute-value test at a node would also produce the second best with the same score. This would result in a always having value 0. ITI was also modified to avoid this caveat.

Table 2 shows how the proposed improvement may lead to speed-up in the above domains. The numbers reported refer to the fraction of the effort expended by the original algorithm that has been expended by the improved algorithm to solve the same problem (a baseline performance of 1 refers to the original algorithm). The results are shown in the form Mean-Value (Standard-Deviation). We report the results with an accuracy of two decimal digits.

Table 2. Results on the amount of computations.

	U-Nodes	U-Attributes	CPU time
kr-vs-kp	0.60 (0.06)	0.55 (0.06)	0.88 (0.05)
mushroom	0.46 (0.03)	0.41 (0.03)	0.78 (0.06)

The quantities (actually, the ratios) reported are:

- **U-Nodes**: The number of times a node had to be considered because its test could be unstable.
- **U-Attributes**: The number of times an attribute had to have its information gain calculated because of the instability of the test at the node where the attribute was available.
- **CPU time**: The CPU time (in seconds) required for an experiment (measured using the *getrusage* system function).

The results confirm that substantial savings are realized, that translate to direct speed-up in the running of the algorithm. However, beyond test stability at a node, other aspects of efficiency in incremental induction must be identified and investigated to examine the apparent differences in the ratios of the **U-Nodes**, **U-Attributes** and **CPU time** quantities between the improved and the original versions of the algorithm. This means that the problem of test stability is not the only bottleneck in the process of incremental induction and that updating the decision tree with all required information about a new instance is quite an expensive task.

Some interesting observations can be made regarding the tentative claims made earlier, about the type of problems where the proposed method would be most effective (we had indicated that higher nodes could be expected to be more stable in their selection of tests than lower nodes and that large data sets would show more pronounced improvement). Note that the results for the **mushroom** domain are more impressive than the ones for the **kr-vs-kp** domain. This is reflected in an examination of the training instances of these domains, as in the **kr-vs-kp** domain there exists an abundance of binary attributes and fewer instances are available.

It should be stressed, that this comparison across experimental domains only serves as a preliminary investigation and simply confirms that there exist many aspects in the problem of efficient incremental induction that merit attention.

5. Discussion

The ID5R (ITI) algorithm has been enhanced so that it takes into account the quality of the available attributes and, thus, saves on the number of times that it evaluates and compares scores of competing attributes. On the basis of these improvements it has been shown to perform better on non-trivial induction problems. We believe that the analysis is representative of the procedure one should employ to evaluate the method with other attribute selection policies, such as the gain-ratio (Quinlan, 1986), the Gini index (Breiman et al., 1984), or the de Mántaras distance (de Mántaras, 1991). We hope that similar results will be established in this direction, in the course of future work.

Note that when a transposition occurs at a given node, all lower nodes may be reconsidered, in the worst case. This does not affect the analysis above, as new nodes generated by the transposition are treated as if they were generated during the normal splitting procedure (we compute a for each node, as soon as it is generated). However, we have not studied the effects of transpositions in greater detail and it is unclear whether they could be used in the analysis to obtain larger values for a.

Note also that the proposed enhancement does not affect the worst- case complexity of ID5R. This happens because it is a heuristic approach that aims to exploit regularities that we expect to exist in most problem domains. Note that any ordering of patterns that makes all visited nodes to be re-considered after each step, will result in worse overall performance, when compared to ID5R. This happens because of the overhead incurred to compute a, while a will always be 0.

The proposed approach may be also compared to the sub-sampling problem (Catlett, 1992; Musick et al., 1993), where the objective is to estimate reliably how much the problem size parameters may be scaled down, without compromising quality. In the context of incremental learning, we define quality as compatibility with ID3 (as outlined in the introduction, where the scope of this paper was set out), but one could also take a broader view.

A special case of performance improvement occurs when the computed value for a (call f the corresponding attribute) is larger than the number of different values of any competing attribute (call it g). In such a case we have guaranteed f's superiority over g for the subsequent a steps, assuming that g places each new pattern at an empty bucket. This extreme-case assumption, however, cannot hold for a steps, but rather for the number of different values for g. This means that before the a-th instance is observed, g will have depleted its pool of possible values and new instances (up to the a-th) will be allocated to already populated buckets. We refer to this situation as "folding". When folding occurs, the value for a is too pessimistic, as maximal score decreases cannot be eventually achieved for competing attributes and the originally best splitting attribute is further strengthened. Preliminary attempts to quantify the effect of folding have not been successful in the sense that the reduction in the number of node checks may not justify the cost of calculating a, for every node. We strongly believe, however, that tighter bounds will unveil whether folding may finally be of any practical value.

A further potential improvement would be to establish a set of values for a, for each node, where the ith component of vector a would refer to the ith best attribute for that node. The

revised algorithm would then need to look at an attribute score according to the value of a component of a. Although this would probably save some more computational steps, it is doubtful whether time savings could be realized, as the additional overhead may be too high.

Another interesting situation arises when an attribute is only marginally better than its competitors. In this case it is unlikely that whichever attribute prevails will also assume a safety margin that will carry it more than one further new instance. If reconsiderations and pull-ups happen often, swapping of "currently best" attributes for some parts of the tree may be experienced, which, in extreme cases, may carry on until the end of the learning process. The applicability of the proposed method in this situation can probably be studied in conjunction with the requirement that the optimal tree (in the sense of ID3-compatibility) be available for the learning of each new training instance. No relevant studies have been conducted, so the above point is offered as speculation only (however, Schlimmer and Fisher (1986) and Utgoff (1989) do discuss the subject of stability of decision nodes during incremental learning).

6. Conclusion

A method of incremental decision tree induction has been presented and evaluated, analytically and experimentally. It was shown to offer significant improvements over an existing technique and research questions related to it have been identified.

Acknowledgments

D. Kalles would like to acknowledge the support of the EPSRC (U.K.) and the NATO studentships' commission of the Greek Government throughout his research. He would also like to thank a colleague, George Paliouras, whose comments on an earlier draft of this paper resulted in several improvements.

The authors would also like to acknowledge the help of Paul Utgoff and of the anonymous reviewers of the Machine Learning journal. Their comments have substantially improved the presentation of this paper.

Notes

1. Note that the original assumption, that a positive instance was available, does not affect the resulting expressions. This supports the generality of the argument.

References

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Brodley, C.E. and Utgoff, P.E. (1995). Multivariate decision trees. *Machine Learning*, 19:45–77.

Catlett, J. (1992). Peepholing: Choosing attributes effectively for megainduction. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 49–54, Aberdeen, Scotland.

Cockett, J.R.B. (1987). Discrete decision theory: Manipulations. *Theoretical Computer Science*, 54:215–236.

Cockett, J.R.B. and Zhu, Y. (1989). A new incremental technique for decision trees with thresholds. *Proceedings of the SPIE*, 1095:804–811.

de Mántaras, R.L. (1991). A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 4:81–92.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172.

Hyafil, L. and Rivest, R.L. (1976) Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17.

Moret, B.M.E. (1982). Decision trees and diagrams. *ACM Computing Surveys*, 14:593–623.

Murphy, P. and Aha, D. (1995). UCI repository of machine learning databases. Machine readable data repository, (ftp to) ics.uci.edu, (cd to) pub/machine-learning-databases, University of California, Irvine.

Musick, R., Catlett, J., and Russell, S. (1993). Decision theoretic subsampling for induction on large databases. In *Proceedings of the 10th International Conference on Machine Learning*, pages 212–219, Amherst, MA.

Naumov, G.E. (1991). NP-completeness of problems of construction of optimal decision trees. *Soviet Physics: Doklady*, 36(4):270–271.

Pagallo, G. (1989). Learning DNF by decision trees. In *Proceedings of the 11th Joint International Conference on Artificial Intelligence*, volume 1, pages 639–644, Detroit, MI.

Quinlan, J.R. (1983). Learning efficient classification procedures and their application to chess end games. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an Artificial Intelligence Approach*, pages 463–482. Tioga Publishing, Palo Alto, CA.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Quinlan, J.R. (1993). *C4.5 Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Schlimmer, J.C. and Fisher, D. (1986). A case study of incremental concept induction. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 496–501, Philadelphia, PA. Morgan Kaufman.

Utgoff, P.E. (1988). ID5: An incremental ID3. In *Proceedings of the 5th International Conference on Machine Learning*, pages 107–120, Ann Arbor, MI.

Utgoff, P.E. (1989). Incremental induction of decision trees. *Machine Learning*, 4(2):161–186.

Utgoff, P.E. (1994). An improved algorithm for incremental induction of decision trees. In W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick, NJ Morgan Kaufmann.

van de Velde, W. (1990). Incremental induction of topologically minimal decision trees. In *Proceedings of the 7th International Conference on Machine Learning*, pages 66–74, Austin, TX.

Received May 3, 1994
Accepted February 27, 1995
Final Manuscript February 21, 1996