Overpruning Large Decision Trees

Jason Catlett
Basser Department of Computer Science
University of Sydney
NSW 2006, Australia

Abstract

This paper presents empirical evidence for five hypotheses about learning from large noisy domains: that trees built from very large training sets are larger and more accurate than trees built from even large subsets; that this increased accuracy is only in part due to the extra size of the trees; and that the extra training instances allow both better choices of attribute while building the tree, and better choices of the subtrees to prune after it has been built. For the practitioner with the common goals of maximising the accuracy and minimising the size of induced trees, these conclusions prompt new techniques for induction on large training sets. Although building huge trees from huge training sets is computationally expensive, pruning smaller trees on them is not, yet it improves accuracy. Where a pruned tree is considered too large for human or machine limitations, it can be overpruned to an acceptable size. Although this requires far more time than building a tree of that size from a correspondingly small training set, it will usually be more accurate. The paper also describes an algorithm for overpruning trees to user-specified size limits; it is evaluated in the course of testing the above hypotheses.

1 Introduction

Algorithms that induce decision trees from examples have been the subject of much machine learning (ML) research in recent years (Quinlan, 1986). As this family of algorithms has developed and propagated, the size and complexity of the training sets attacked has grown. For example, Dietterich, Hild & Bakiri (1990) report using decision tree techniques on 140 000 instances concerning the pronunciation of English. The large corporate databases that could potentially be used for commercial applications such as retail credit assessment (Carter & Catlett, 1987) are commonly much larger. Catlett (1991c) describes an application using hundreds of thousands of examples from the Space Shuttle; NASA's archives of this flight data contain more than 150 million training examples.

Decision trees induced from noisy data have a well-documented tendency to "overfit" the training data (Quinlan, 1986). The lower nodes in the tree, built from few examples, are often testing irrelevant attributes, fitting the noise in the training data. Such spurious nodes lower accuracy on unseen data. Two common ways of avoiding this are to impose some criterion checking the relevance of attributes before building each node, and to go back after the tree has been built and remove subtrees judged not to be contributing a sufficient increase in accuracy for their size.

Even if all redundant nodes have been perfectly pruned, a tree may still be considered too large, for reasons other than accuracy. The most prominent reason is that large decision trees are incomprehensible. Michie (1987) argues strongly from many years of knowledge engineering projects that the AI goal of producing concepts human experts can understand dictates that the size of decision trees must be kept within a fairly small "human window." Comprehensibility is of course not determined by size alone; structure is very important, as Shapiro (1987) demonstrated. Even when the size and structure of two concepts are the same, human beings may find one more comprehensible. But size is the most basic parameter, and it is easy to measure objectively.

Beyond the bounds of comprehensibility, limitations of resources may also force limits on the size of trees. A tree may be required to fit within a certain amount of memory, or to classify examples within a maximum amount of time.

Given that it is sometimes desirable to *overprune* trees, sacrificing some accuracy in order to reduce size, how much is enough? In their evaluation of their CN2 induction algorithm, which aims at building small sets of rules, Clark & Niblett (1989) comment on the tradeoff between accuracy and size (or *complexity*). "We do not attempt to combine measures of accuracy and complexity... as such a combining function is dependent on the purpose for which the rules are to be used. For example, a 1% fall in predictive accuracy may or may not justify a 10% decrease in rule complexity, depending on the application." Thus an overpruning algorithm should

allow tailoring of the size and accuracy of a tree according to the particular economics of the application.

This paper investigates the effect on size and accuracy of building and pruning trees from large training sets. Section 2 states and illustrates the hypotheses to be tested. Section 3 gives a brief summary of the existing techniques for building and pruning trees, then describes new pruning and overpruning algorithms that are based on them. Section 4 briefly describes the domains used for testing, specifies the experiments and presents their results. Section 5 discusses further and related work.

2 The hypotheses to be tested

Before stating the hypotheses, some graphical illustrations of the relationships between training set size, error rate and tree size may help clarify their motivation.

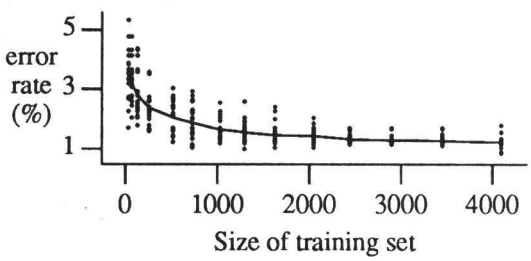

Figure 1: Hyperthyroid Learning Curve (Linear Version)

Figure 1 is an example of the most common graph in ML papers, often called a *learning curve*. The domain of the training instances is the hyperthyroid data described in (Quinlan, Compton, Horn & Lazarus, 1987). The (piecewise linear) curve joins the average error rate of pruned decision trees built from various fixed sizes of training set. Here the individual results of each of the twenty iterations are also plotted as dots. At each iteration the training set starts with 32 examples, and training instances are added without replacement in batches of the various predetermined sizes, with a new tree grown and assessed after each batch. Unsurprisingly, the marginal improvement in error from additional training instances diminishes with the size of set. It is not easy to see whether the marginal improvements have vanished by the right hand side of the graph, but transforming both axes to log scales makes the picture clearer. Extrapolation of the roughly straight line in Figure 2 suggests that improvements in accuracy may be possible, but at the cost of processing much larger training sets. The graph does indicate whether the comparatively small improvements in accuracy being gained towards the right of the graph are statistically significant; this is tested as the first hypothesis, H1: Very large training sets yield significantly more accurate trees than subsets half their size. These experiments used randomly chosen subsets, some selective strategies such as removing duplicated examples and balancing class frequencies with stratified samples are investigated in (Catlett, 1991b).

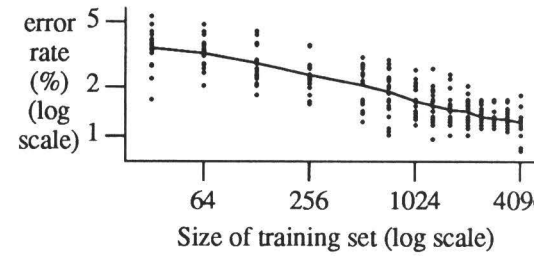

Figure 2: Hyperthyroid Learning Curve (Log Version)

Turning from accuracy to size, Figure 3 plots the size of the pruned trees on a linear scale, showing a roughly linear relationship between the size of the training set and the size of the tree. This is bad news for the goal of smaller trees. The hypothesis that the size of pruned trees continues to grow as the training set becomes very large is formulated as H2: Very large training sets yield significantly larger pruned trees than subsets half their size.

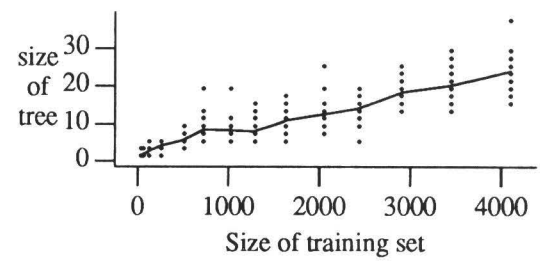

Figure 3: Growth Of Tree Size (Hyperthyroid)

To see the tradeoff between tree size and error rate, the y-values from Figures 2 and 3 can be presented as a bivariate scattergraph, hiding the training set sizes. Figure 4 shows tree size increasing exponentially as a function of accuracy (decreased error rate), suggesting that if larger training sets are used to reach a goal of greater accuracy, this quickly subverts the goal of small size. The ratio of the computational cost of induction to increased accuracy is even worse; this problem is illustrated in (Catlett, 1991c) and attacked in (Catlett, 1991a).

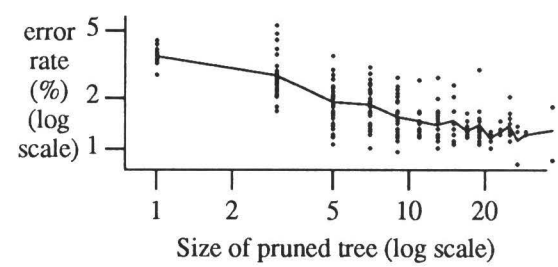

Figure 4: Accuracy Versus Tree Size (Hyperthyroid)

A practitioner faced with this analysis and a practical induction task with a huge amount of data might

reasonably ask how the most accurate tree can be built within given size and time constraints. To design algorithms to meet this requirement requires more knowledge about why the error rates of the larger trees obtained from larger training sets are more accurate. There are three possible reasons, which are exhaustive but not mutually exclusive; they comprise the last three hypotheses. The first is the most obvious explanation. H3: The greater size of the trees built from larger training sets improves their accuracy. H4: The better choice of attributes made while building trees from larger training sets improves their accuracy. H5: Larger sets of examples used for pruning trees provide greater accuracy.

Empirical evaluation strongly corroborates all the hypotheses, motivating new algorithms and methods that are used in the experiments evaluating these hypotheses. H1 motivates the use of very large training sets to improve accuracy. H2 shows that some method of overpruning the resulting trees will be needed to impose a limit on tree size. H3 shows that this will incur a cost in accuracy, but H4 and H5 show that the accuracy will still be superior to trees built from just a small set. H4, along with information on the high cost of evaluating attributes on large training sets, motivates the development of more efficient and effective methods of choosing attributes during the construction of trees, such as (Catlett, 1991a). H5 prompts the simple idea (described in the next section) of using a larger set for pruning than was used for building.

3 Tree building and pruning

The program used in these experiments for building and pruning decision trees was C4.5, implemented in C and distributed by Quinlan. The basic algorithm at the core of C4.5 is ID3. For a comprehensive introduction to ID3 see (Quinlan, 1986). Various methods of pruning are described in (Quinlan, 1987); C4.5 uses a variation on *pessimistic pruning*. This method uses a statistical adjustment to estimate the error rate on unseen data likely to be achieved by each subtree of the tree. A subtree S at a node is replaced by a leaf if S is estimated to have an error rate higher than the rate estimated for a leaf with the majority class in that position. Similarly, C4.5's pruning algorithm also examines the largest branch B of each node S. (B is the subtree immediately below the root of subtree S that represents the most examples, not necessarily the one with the most nodes.) If the estimated error rate of the subtree where S is replaced by B is lower than the estimate for S as it stands, the replacement goes ahead and the sibling branches are discarded. One might ask in passing why C4.5 considers only the largest branch. Intuition suggests that this branch is the one most likely to give an improvement, and that searching all of the branches doing a recursive estimates would take a lot more time. Experiments on the domains described in this paper showed that although branches other than the largest occasionally gave lower estimates, searching all the branches rarely resulted in a significantly lower error rate for the pruned tree. The time taken was greater, but not by a large factor.

Pessimistic pruning has the advantage over other pruning methods such as cost-complexity pruning (Breiman, Friedman, Olshen & Stone, 1984) that it does not require that the set of examples used for pruning be distinct from the set used to build the tree. Where training instances are scarce, it makes sense to use them all during the tree-building, and then to use them again in pruning. But where resource constraints demand that only a sample of the available training instances can be used to build a tree, it is possible (and according to H5 may be beneficial) to use extra examples to prune it. (Pruning a tree requires far less time than building it.) This simple technique requires no change to the pruning algorithm and only a minor change to the code to implement, but as shown in Table 3 in Section 4 below, it gives significantly greater accuracy.

The next technique builds on the pessimistic pruning methods of C4.5, aiming to *overprune* a tree to a user-specified size. It not always possible to prune a tree to exactly the specified size: for example, a tree of three nodes cannot be pruned to two. But the algorithm is guaranteed to produce a tree of at most the specified number of nodes. The algorithm assumes that the tree has already been pruned, but the extension to underpruning unpruned trees is straightforward. This could be useful if it were known that the regular pruning algorithm was pruning too aggressively in a domain.

The basic idea of the overpruning algorithm is to remove those nodes that result in the least increased error estimate, while still taking the total size of the tree within the limit. (An early version on the overpruning algorithm, called OP1, removed the parts of the tree that represent the least number of examples regardless of the error estimates.) Finding the set of subtrees that gives the lowest total estimated increased error is a difficult combinatorial problem, so two iterative simplifications of the optimal solution were used. The first (OP2) searches the tree at each iteration for the node that gives the lowest estimated increase in error; that node is pruned before the next iteration. This method favours the parents of leaves, and sometimes results in a lot of nibbling at the fringes of the tree, when biting off a big subtree would have given a lower increase in error. To counter this the estimate is adjusted by dividing by the size of the subtree being considered. (Actually this may unduly favour lobbing off a subtree larger than necessary to get the tree below the required limit, so the denominator used was the maximum of the size of the subtree and the remaining reduction required.) This version is called OP3. Finally, the same technique applies to replacing nodes with branches as well as leaves; the version that does both is called OP4. Empirical evaluation showed the accuracy of trees given by each version superior to its predecessor;

the results reported below are for OP4.

The current implementation is less efficient than necessary; it rescans the entire tree at each iteration. It would be better to store the statistics at the nodes and only recompute those parts of the tree that change. Even with this gross disadvantage, the overpruning program is typically cheaper to run than the tree building program.

A different way to achieve overpruning, but not to a parameterised size, is to bias the estimated error from subtrees and leaves so as to favour leaves. I experimented with a parameter that specifies the minimum ratio of these estimates, but was able to attain only very crude control of tree size.

A possible advantage of iterative methods of overpruning is that at each iteration statistics can be produced on the size and estimated error rate of the intermediate trees. Based on this information a developer might decide to use one of these less severely pared trees instead. The error estimates for trees that C4.5's pruning algorithm produces are often very accurate, but if more precision were required they could be recalibrated by assessing the error rate of some of the trees on a separate set of examples and adjusting the other estimates accordingly.

4 Description of domains and experiments

Ten noisy domains of a suitable size for these experiments were found. The first three are synthetic in the sense that the data are generated by programs; the remaining seven are natural domains of interest to people other than computer scientists. The last five are various families of thyroid disorders (Quinlan *et al.* , 1987), but can be considered separate domains. Space does not permit a full description of the domains; this appears in (Catlett, 1991b). Two of them are time series data from bio-medical engineering applications, concerning ischemia (Oates, Cellar, Bernstein, Bailey & Freedman, 1989) and sleep disorders (Ray, Lee, Morgan & Airth-Kindree, 1986); others are familiar from the ML literature: the Waveform generator from (Breiman *et al.*, 1984), and Othello generator from (Utgoff & Heitman, 1988). The diff domain simply classifies pairs of real numbers according to the sign of their difference.

All the experiments followed the same procedure and are reported in the same format. Since the number of examples in the datasets varies from just over 5 000 to over 7 000 (or an unlimited number in the case of the synthetic domains), and a distinct test set of at least 2 000 examples was reserved, this left two sizes for the training set: 3 000 in the case of heart, othello and sleep, and 5 000 for the others. Training examples from synthetic domains were synthesised anew with a different seed for each of the iterations described below, rather than using a fixed set of 7 000 examples generated once.

Each experiment compares two values of a single parameter (error rate or tree size) obtained from two different treatments (called A and B) of an series of 20 constructed training sets. For example, in the first experiment treatment B uses the full training set to learn from, and A uses only half the set. For each of the 20 iterations, a training set and a test set are constructed by splitting the data randomly according to the sizes specified above. The parameter is measured by evaluating each pair of trees on the test set. The difference of this pair of parameters is averaged over all iterations; the tables report the standard deviation of this difference, as well as the average of the trees from treatment A and B. The aim is to test the null hypothesis that there is no difference observed on the parameter between treatments A and B; this is done with a two-sided Student t-test at the 5% level. (Whether this "statistically significant" difference is large enough to be significant to a client obviously depends on the application.) In the tables below, an entry of "A" under the heading "NH" indicates the null hypothesis is accepted. An entry of "R+" indicates that the difference is positive (i.e. A is significantly higher than B), and an entry of "R–" indicates that the null hypothesis is rejected and that A is significantly lower than B.

Table 1 shows the result of testing H1, that very large training sets yield significantly more accurate trees than subsets half their size. Treatment A uses a randomly sampled 50% of the training set; B uses the full training set. The result is unanimously "reject+", indicating that A has a higher error rate than B, in other words, the full training set (B) gives more accurate trees, corroborating the hypothesis.

Domain	A	B	sd(A-B)	NH
wave	25.93	24.80	1.28	R+
diff	2.27	1.58	0.31	R+
othello	20.79	16.77	1.39	R+
heart	4.39	2.89	0.69	R+
sleep	27.62	25.24	0.98	R+
hyper	1.29	1.08	0.16	R+
hypo	0.70	0.60	0.14	R+
binding	2.57	2.11	0.39	R+
replace	1.76	1.41	0.44	R+
euthy	0.91	0.56	0.22	R+

Table 1: Error rates: B&P-50% (A) vs B&P-100% (B)

The second experiment uses the same treatments, but the dependent variable is the size of the trees built. This tests H2, that size of pruned trees continues to grow as the training set becomes very large. The statistical test shows only that the trees have almost certainly grown by some positive amount, but the estimated means in Table 2 indicate a growth of about 30-90%, strongly corroborating the hypothesis. Tree size appears to be a roughly linear function of training set size in these

domains, although on the noise-free Shuttle data described in Catlett (1991c) the function more resembled the cube root.

Domain	A	B	sd(A-B)	NH
wave	326.90	625.00	51.07	R–
diff	59.80	87.70	8.93	R–
othello	208.90	352.50	24.53	R–
heart	45.90	74.20	12.13	R–
sleep	135.20	234.70	25.10	R–
hyper	15.00	26.10	6.31	R–
hypo	19.10	27.70	5.92	R–
binding	30.90	46.90	9.99	R–
replace	22.10	30.70	8.39	R–
euthy	28.00	37.40	4.95	R–

Table 2: Tree size: B&P-50% (A) vs B&P-100% (B)

The next experiment tests H5, that a larger training set allows better pruning of the tree. Table 3 shows error rate as the dependent variable; Table 4 shows tree size.

Domain	A	B	sd(A-B)	NH
wave	29.45	26.42	1.46	R+
diff	5.18	4.47	0.76	R+
othello	33.25	29.27	1.76	R+
heart	7.93	7.33	0.74	R+
sleep	32.88	29.95	1.61	R+
hyper	2.15	1.95	0.34	R+
hypo	1.34	1.22	0.21	R+
binding	4.28	3.79	0.60	R+
replace	2.24	2.05	0.28	R+
euthy	1.92	1.60	0.44	R+

Table 3: Error rate: B&P-10% (A) vs B-10%;P-100% (B)

In both treatments A and B the trees were built on 10% of the training set, but B was given the full training set for the pruning, whereas A was pruned on the same 10%. The unanimous result on error rate corroborates Hypothesis 5. Table 4 shows that in many domains the smaller trees were obtained by pruning on a larger set. In two domains, othello and diff, they were larger.

Domain	A	B	sd(A-B)	NH
wave	74.60	35.00	12.30	R+
diff	25.80	30.80	5.93	R–
othello	55.20	62.90	14.86	R–
heart	16.50	14.80	2.70	R+
sleep	47.30	36.50	9.94	R+
hyper	6.10	6.20	2.94	A
hypo	9.30	8.40	1.21	R+
binding	9.80	9.30	5.50	A
replace	12.60	11.90	2.54	A
euthy	11.80	11.70	2.55	A

Table 4: Tree size: B&P-10% (A) vs B-10%;P-100% (B)

Table 5 concerns H4, that the larger training set allows a better choice of attributes. To test this both A and B are given a full set for pruning, but for the tree building, A is given only 10% of the examples, whereas B is given the full set. To eliminate differences due to the larger size of B's trees, each tree built by B was overpruned to the same size as the corresponding tree from A. The hypothesis held for all domains except euthyroid.

Domain	A	B	sd(A-B)	NH
wave	26.42	25.14	1.11	R+
diff	4.47	3.97	0.53	R+
othello	29.27	25.87	1.34	R+
heart	7.33	5.90	1.11	R+
sleep	29.95	27.56	1.52	R+
hyper	1.95	1.56	0.67	R+
hypo	1.22	0.83	0.23	R+
binding	3.79	3.02	0.71	R+
replace	2.05	1.75	0.33	R+
euthy	1.60	1.54	0.37	A

Table 5: Error rate: B-10%;P-100% (A) vs overpruned (B)

The results of two other experiments warrant reporting without the extra tables. Varying the previous experiment so that A prunes only on the same 10% it uses for building tests a different hypothesis, that trees built from a small sample are less accurate than trees built from a large set and overpruned to the same size as the small trees. The results unanimously corroborated this hypothesis; the difference in error rates were mostly a factor of two or three times those in Table 5. The uncontentious hypothesis that additional size of trees can improve their accuracy (H3) was tested by an experiment similar to that of Table 5, except that A was changed to growing and pruning on the full training set (the same as B without overpruning). The larger trees from A were significantly more accurate in all domains except wave. This exception suggesting that pruning may not be aggressive enough in this domain.

5 Discussion and Conclusion

These experiments corroborate the five hypotheses stated in Section 2, showing where additional accuracy can be squeezed out of learning algorithms. They also demonstrate the utility of the two new techniques used in the experiments: pruning on a larger set, and overpruning. Improvements that are statistically significant can be hard to obtain when error rates are already below 1%, and in some domains improvements of several percentage points were obtained. These techniques let the practitioner move closer to the fundamental goals obtaining small and accurate concepts from learning systems.

Several avenues for further work are open. A close evaluation of the efficiency and effectiveness of the overpruning algorithm might lead to improvements, although the construction of the large tree in the first place is more important to the goal of conserving resources. A method of speeding up this step on very large training sets sets is described in (Catlett, 1991a). The conversion of overpruned trees to production rules is obviously possible, but the control over the size of the rule base will not be directly specified. An evaluation of the tradeoff between accuracy and comprehensibility to humans would be of great interest to those concerned with synthesising knowledge from large data bases. Although comprehensibility to human experts degrades as the number of rules or nodes increases, the tolerances around a critical point are probably not as tight as a single node, so this batch-style overpruning algorithm may be less appropriate than some kind of interactive pruning system, analogous to the tree building of Shapiro's Interactive ID3 (Shapiro, 1987). Presenting an expert with a series of trees in order of increasing complexity may enhance comprehension.

Comparisons with a wide variety of existing ML algorithms could be made on the basis that overpruning is one way of buying additional accuracy by spending CPU time (assuming that training instances are abundant and cheap). Other ways of doing this include spending more time on the tree building, for example by attempting to build a minimal tree, as in (Van de Velde, 1990), and improving the attributes by constructive induction (Rendell, 1989). Such investigations should further understanding about the relationship between the fundamental parameters of learning time and concept size and accuracy, and aid the development of techniques for best achieving specific goals for these parameters.

Acknowledgments

Many people deserve thanks for providing data: Ben Freedman and John Oates of Royal Prince Alfred Hospital for the heart data, Ross Quinlan for the thyroid data, Sylvian Ray for the sleep data, and Paul Utgoff for the othello data. Thanks are also due to Ross Quinlan for his advice and suggestions, particularly concerning some of the variations of the overpruning algorithm, and for supplying code for C4.5. Discussions with Donald Michie motivated me to focus on size and comprehensibility. This work was supported in part by a grant from the Australian Research Council.

References

Breiman *et al.* 1984. Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

Carter & Catlett 1987. C. Carter and J. Catlett. Assessing Credit Card Applications Using Machine Learning. *IEEE Expert 2(3)* (Fall 1987) pp. 71-79, IEEE Computer Society.

Catlett 1991a. J. Catlett. *Choosing attributes through peepholes.* Basser Department of Computer Science, University of Sydney, 1991.

Catlett 1991b. J. Catlett. *An evaluation of techniques that select subsets of training sets.* Basser Department of Computer Science, University of Sydney, 1991.

Catlett 1991c. J. Catlett. Megainduction: a test flight. In Birnbaum, L., and Collins, G. (Ed.) *Machine Learning: Proceedings of the Eigth International Workshop.* San Mateo, CA. Morgan Kaufmann.

Clark & Niblett 1989. Peter Clark and Tim Niblett. The CN2 Induction Algorithm. *Machine Learning 3,4* (1989).

Dietterich *et al.* 1990. Thomas G. Dietterich, Hermann Hild, and Ghulum Bakiri. A comparative study of ID3 and backpropagation for English text-to-speech mapping. In Porter, B., and Mooney, R. (Ed.) *Proceedings of the Seventh International Conference on Machine Learning.* Austin, Texas. Morgan Kaufmann.

Michie 1987. D. Michie. Current Developments in Expert Systems. In Quinlan, J.R. (Ed.) *Applications of Expert Systems.* Glasgow. Turing Institute Press with Addison Wesley.

Oates *et al.* 1989. J. Oates, B. Cellar, L. Bernstein, B.P. Bailey, and S.B Freedman. Real-time detection of ischemic ECG changes using quasi-orthogonal leads and arificial intelligence. In *Proceedings, IEEE Computers in Cardiology Conference.*

Quinlan 1986. J. R. Quinlan. Induction of Decision Trees. *Machine learning 1,1* (1986) pp. 81-106.

Quinlan 1987. J. R. Quinlan. Simplifying Decision Trees. *International Journal of Man-machine Studies 27* (1987) pp. 221-234.

Quinlan *et al.* 1987. J.R. Quinlan, P.J. Compton, K.A. Horn, and L. Lazarus. Inductive knowledge acquisition: a case study. In Quinlan, J.R. (Ed.) *Applications of Expert Systems.* Glasgow. Turing Institute Press with Addison Wesley. pp. 155-173

Ray *et al.* 1986. S.R. Ray, W.D. Lee, C.D. Morgan, and W. Airth-Kindree. Computer sleep stage scoring: an expert systems approach. *International Journal of Bio-Medical Computing 19* (1986) pp. 43-61.

Rendell 1989. L. Rendell. Comparing systems and analysing functions to improve constructive induction. In Segre, A. (Ed.) *Proceedings of the Sixth International Workshop on Machine Learning.* Ithaca, New York. Morgan Kaufmann. pp. 461-464

Shapiro 1987. Allen D. Shapiro. *Structured Induction in Expert Systems.* Turing Institute Press, Glasgow, 1987.

Utgoff & Heitman 1988. P.E. Utgoff and P.S. Heitman. Learning and generalizing move selection preferences. In *Proceedings of the AAAI symposium on computer game playing.* pp. 36-40

Van de Velde 1990. Van de Velde, Walter. Incremental induction of topologically minimal trees. In Porter, B., and Mooney, R. (Ed.) *Proceedings of the Seventh International Conference on Machine Learning.* Austin, Texas. Morgan Kaufmann. pp. 66-74