Overcoming Fragmentation in Decision Trees Through Attribute Value Grouping

K. M. Ho and P. D. Scott

Department of Computer Science,
University of Essex,
Colchester, CO4 3SQ, UK
hokokx@essex.ac.uk and scotp@essex.ac.uk

Abstract. In this paper we investigate several methods for producing smaller decision trees by reducing fragmentation through the use of methods that lower the mean branching factor. All the methods considered achieve this goal by grouping the values that each attribute may take. We show how such grouping may be carried out by using either top-down iterative splitting or bottom-up iterative merging. Such methods may be applied either globally at the onset of tree construction or locally whenever a new node is considered. We also compare two approaches to assessing the quality of such attribute value groupings: information gain ratio, as employed in C4.5, and a combination of χ^2 and Cramer's V. The results of a comparative study of eight methods show that a top-down global method, using χ^2 and Cramer's V, produces consistently smaller tree sizes without loss of accuracy or computation time. These findings may be of considerable practical importance in data mining since it is widely recognised that smaller trees are much easier to understand.

1 Introduction

Decision tree induction ([1],[13]) has become firmly established as one of the most widely used data-mining techniques. Unfortunately experience with real world data sets has shown that existing methods may construct very large decision trees, that may be of limited practical utility. Any method for reducing the size of decision trees without sacrificing classification accuracy would be of considerable practical importance in mining large real world data sets.

In this paper we introduce and compare a number of techniques for restricting the size of decision trees by reducing their average branching factors. We begin, in Section 2, with a review of the origins of over-complexity in decision trees and existing methods for simplifying them. In particular we examine the problem of unnecessary fragmentation of the sample space that arises through forming distinct subtrees for every value of a nominal attribute. In the following section we examine several different approaches to grouping attribute values to reduce such fragmentation. Section 4 presents a series of experimental comparisons of eight alternative methods for grouping the values of nominal attributes. Seven of these are novel, while the eighth is the method provided as an option in C4.5 [13]. We conclude that a top-down global method is the most effective technique for reducing fragmentation in decision trees.

2 Controlling Decision Tree Size

One important advantage of using decision tree techniques is that this form of classification function can be easily understood, provided that the tree produced is reasonably small. In recent years, because of the rapidly rising interest in data mining, these techniques have been used with ever larger sets of training data. Unfortunately large training sets may lead

to large decision trees containing many thousands of nodes. Oates and Jensen [11] have presented evidence that tree size often increases linearly with training set size. Large trees may prove excellent classifiers but they are so difficult to comprehend that they provide little useable information about the patterns that exist in the data. Developing methods of producing smaller decision trees without sacrificing classification accuracy is therefore an important research problem whose solution will have widespread practical application.

The size of a decision tree, defined as the number of nodes it contains, will depend on both the maximum depth and the mean branching factor of its nodes. It follows that if we are to develop decision tree construction techniques that produce smaller trees we must use one or both of two possible approaches:

1. Restrict the maximum depth of the tree and its subtrees.
2. Restrict the branching factor of the non-terminal nodes.

Each of these approaches could be applied either *a priori* or *a posteriori*. An *a priori* restriction is applied to constrain or eliminate a portion of the tree before it has been built, whereas an *a posteriori* restriction is used to remove a portion of the tree that has already been constructed. Depth restriction is often known as *pruning*.

2.1 Overfitting and Pruning

One important cause of excessive decision tree size is overfitting. It manifests itself in decision tree construction as excessively deep trees. It can be limited by restricting the depth to which the tree is allowed to grow, but the use of stopping criterion, sometimes called *pre-pruning*, may be ineffective in detecting when the procedure has begun to model characteristics peculiar to the sample [4]. Consequently the majority of researchers (e.g. [13]) favour some form of *post-pruning* in which apparently useless subtrees are removed from a complete decision tree. Breslow and Aha [2] have provided a comprehensive review of a wide range of pruning techniques.

In principle therefore, effective pruning is a technique for reducing tree size, without sacrificing classification accuracy, by eliminating those parts of the original tree that arise through overfitting. In practice, pruning is used extensively even at a small cost in accuracy because of the greater accessibility of smaller trees.

2.2 Fragmentation and Attribute Value Grouping

Overfitting leads to decision trees that are too deep. In contrast, a *fragmentation* occurs when a decision tree is too broad: that is, when nodes have more branches than would be necessary to produce the best possible classification accuracy. The following simple example shows how fragmentation is likely to arise frequently with the majority of decision tree construction procedures.

Suppose a decision tree program attempts to build a tree to recognise some class, C, using a training set made up of feature vectors of nominal attributes $A_1, A_2, ..A_k$. Suppose also that the class C is in fact defined thus:

$$C = V_{2,1} \wedge (V_{5,2} \vee V_{5,4}) \wedge V_{8,3}$$

where $V_{i,j}$ denotes that attribute A_i takes its jth value. When a typical decision tree program selects an attribute to construct a new node it will create a separate branch for each of the possible values for that attribute. Thus, assuming as an example that each attribute has four possible values, if the program selects attributes in the order A_5, A_2, A_8 it will construct a tree like that shown on the left of Figure 1. The entire subtree associated with $V_{5,2}$ is repeated

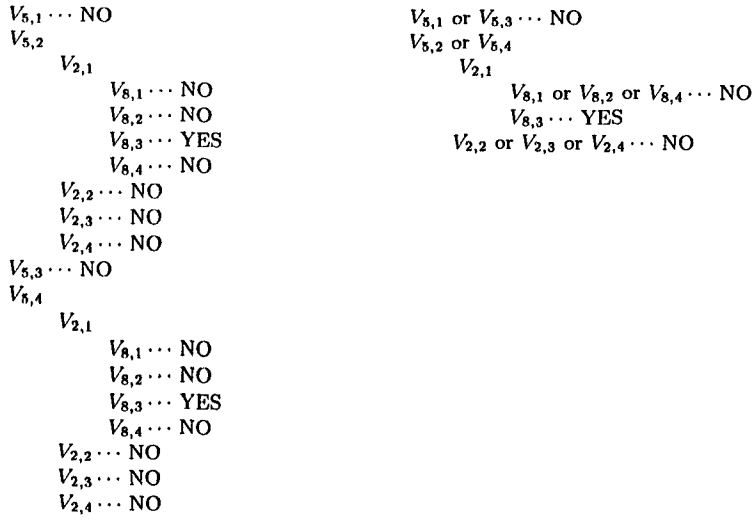

Fig. 1. Alternative decision trees for an example discussed in the text. Left: Without attribute value grouping. Right: With attribute value grouping

in node $V_{5,4}$. This immediately suggests that a much simpler tree could be constructed if the values of attributes could be grouped and then branches created for each group of values rather than for each value. Such a tree is shown on the right of Figure 1.

It is immediately obvious both that the tree without attribute value grouping is considerably larger and exhibits significant fragmentation in that it partitions the example space into 16 regions. In contrast the tree incorporating attribute grouping would produce identical classifications but only divides the example space into 4 regions. Clearly such fragmentation can have a major effect on tree size. In addition it may impair the classification accuracy of the resultant decision tree because the set of training examples must be distributed across more subtrees and hence will provide proportionally less reliable evidence for each.

Pruning will not help to solve this problem: removing any part of the larger tree will seriously affect its classification accuracy. The obvious remedy would be for the program to consider all possible groupings of an attribute's values when searching for the best attribute for a new node, and hence avoid producing more branches than necessary. Unfortunately the number of possible groupings rises very rapidly with the number of values an attribute can take, and hence exhaustive consideration of all groupings is often infeasible.

One solution to this difficulty is to restrict the number of groups. The original CLS program [8] always divided the attribute values into two groups; one containing only the best value and the other containing all the rest. Both CART [1] and Assistant-86 [3] also produce binary trees but permit several values in each group. Fayyad and Irani [6] have produced evidence that such binary groupings are likely to lead to smaller trees than k-way trees, but Kononenko [10] has produced a counter example to the stronger version of this hypothesis. Quinlan has adopted a different approach in C4.5 [13]. If the $-s$ option is selected then the program will search for the best grouping of an attribute's values using a form of agglomerative hierarchical clustering [5]. Such methods are essentially bottom-up hill-climbing procedures in which groups are repeatedly merged. C4.5 continues the process until any further merging would lead to significant degradation of gain ratio.

3 Approaches to Attribute Value Grouping

The development of attribute value grouping procedures to combat fragmentation appears to have attracted much less interest than the problem of devising pruning methods to combat overfitting. There is therefore considerable scope for research work whose goal is to develop alternative grouping procedures and establish how effective they are in decision tree construction.

3.1 Local or Global Groupings

All of the methods referred to in the previous section perform attribute grouping *locally*. They search for the best groupings of each attribute's values every time they select the best attribute for a proposed new node. Although such local groupings can be more finely tuned to the local region of the example space, this approach suffers from two disadvantages: it is potentially costly since the grouping operation is performed many times, and many of the groupings will be based on small samples and hence may not reflect the characteristics of the parent population. We therefore decided to investigate the possibility of developing procedures that carried out attribute value grouping *globally*. Such procedures partition each attribute's values once only, before tree construction begins, using the entire training sample.

Those methods that form only two groups of attribute values are not suitable for global grouping, since they would not be able to form three or more groups under any circumstances. This problem does not arise when they are applied locally since the same attribute may be subject to several successive groupings. Our investigations have therefore concentrated on methods that can partition a k-valued attribute into any number up to k groups.

3.2 Bottom-Up or Top-Down Grouping

As we noted above, C4.5 includes an option that evokes attribute value grouping using a hill-climbing bottom-up method applied locally. This suggests the possibility of a top-down approach that carries out repeated splitting. One such procedure would operate as follows:

1. Create an $m \times n$ table, where m is the number of class labels and n is the number of possible values of the attribute. Let cell$[i, j]$ of the table contain the number of training examples in class i that had the jth attribute value.
2. Identify the modal class for each column and form groups of columns that share the same modal class. There will be at most m such column groups.
3. Apply binary splitting repeatedly to each of these column groups until a point is reached at which either no more splitting is possible or no split satisfies some criterion function (discussed below).

3.3 Group Formation Criteria

Top down methods require a criterion to determine whether splitting a group is worthwhile; bottom-up methods need one to decide whether a pair of groups should be merged. In either case we require a function which will allow us to compare the quality of the representation with and without a particular grouping transformation.

The $-s$ option of C4.5 [13] uses the information gain ratio both to select the best grouping and to determine the point at which merging should stop. In earlier work, Quinlan [12] experimented with χ^2 as a stopping criterion for tree construction but abandoned it because of uneven results. We have had similar experiences using χ^2 as a grouping criterion: we

encountered difficulties in setting thresholds that were appropriate for both large and small sets of training data.

This difficulty is not particularly surprising. The χ^2 test measures the statistical significance of an association between two variables, not its strength. Even the weakest association will appear significant in a large enough data set. We therefore experimented with Cramer's V, a standard statistical measure of the strength of association between two nominal variables [7], defined as:

$$Cramer'sV = \sqrt{\frac{\chi^2}{N(L-1)}} \tag{1}$$

where L is the smaller of the number of rows and columns in 2-way table plotting values of one variable against those of the other.

Our initial experiments with Cramer's V also led to uneven results. This again is unsurprising. Cramer's V tells us whether a relationship is weak or strong if present but not whether there is sufficient evidence to conclude it exists. We therefore decided to use a criterion based on both χ^2 and Cramer's V: the former establishes that an apparent association exists while the latter determines if it is strong enough to be of any interest. As the results reported below show, this combination proved very satisfactory when we used a threshold of $p = 0.01$ for the χ^2 test, and 0.1 for Cramer's V.

	Data Sets	Cases	Classes	2	3	4-5	6-10	11-20	> 20
ad	adult	32562	2	2	1	1	5	3	1
b5	bhps5000	10265	2	3	15	16	59	17	0
b8	bhps8000	10265	2	3	16	16	58	17	0
sm	bhpssmoker	10265	2	6	17	13	57	17	0
bc	breast-cancer	286	2	3	2	0	2	2	0
bs	balance-scale	625	3	0	0	4	0	0	0
dna	DNA-nominal	3186	3	0	0	60	0	0	0
sp	splice	3190	3	0	0	0	60	0	0
car	car	1728	4	0	3	3	0	0	0
nur	nursery	12961	5	1	4	3	0	0	0
soy	soybean-large	683	19	15	14	5	1	0	0

Table 1. Characteristics of the data sets used in evaluations. The columns labelled with numbers indicate the number of values the nominal attributes could take.

3.4 Eight Procedures for Grouping Attribute Values

In this section we have seen that grouping procedures can be used locally or globally, can use a top-down or bottom-up strategy, and can employ either information gain ratio or χ^2 plus Cramer's V as the grouping criterion. Taken together the various combinations define eight distinct methods for attribute grouping. We will use a simple notation to name these in which TD-CC-G denotes a globally applied top-down method using χ^2 plus Cramer's V as its grouping criterion, while BU-IG-L denotes a locally applied bottom-up method using information gain ratio as grouping criterion.

4 An Experimental Comparison

In order to compare all eight attribute value grouping procedures we compared their performance on eleven data sets. Eight of these are well known and were taken from the UCI Repository. Because we wanted to investigate how the grouping procedures would perform

using data sets that had a large number of nominal attributes, we created three more data sets using data from the 1991 wave of BHPS [1] , a large social science survey carried out annually in the UK. In *b5* and *b8* the variable to be predicted is income below 5K and below 8K respectively; in *sm* it indicates whether or not the respondent smokes. All continuous attributes are pre-discretized into categorical groups, therefore, all procedures receive the data as nominal (or discretized) attributes.

We used C4.5 (Release 8 [13],[14]) without the $-s$ attribute grouping option to provide a control, while setting the $-s$ flag provided the BU-IG-L method. The remaining seven methods were implemented by modifying the C4.5 code with different grouping procedures. The code to run the experiments and the generation of training and test datasets were taken from the MLC++ library [9].

Table 2 shows the mean classification accuracies obtained for all nine programs with each data set, while Table 3 shows the mean size of the trees constructed. Limitations of space

	C4.5	Local							
	k-way	BU-IG-L		BU-CC-L		TD-IG-L		TD-CC-L	
	accuracy	accuracy	ratio	accuracy	ratio	accuracy	ratio	accuracy	ratio
ad	86.02 ± 0.33	86.24 ± 0.22	1.00	86.18 ± 0.29	1.00	83.08 ± 0.56	0.97	82.90 ± 0.53	0.96
b5	83.51 ± 0.31	80.87 ± 0.46	0.97	82.38 ± 0.35	0.99	79.13 ± 0.52	0.95	81.39 ± 0.16	0.97
b8	85.06 ± 0.38	82.53 ± 0.27	0.97	83.60 ± 0.69	0.98	77.94 ± 3.60	0.92	83.11 ± 0.58	0.98
sm	72.60 ± 0.47	67.63 ± 0.48	0.93	71.96 ± 0.44	0.99	67.88 ± 1.74	0.93	71.64 ± 0.25	0.99
bc	71.00 ± 2.25	68.20 ± 2.38	0.96	70.65 ± 2.83	1.00	65.75 ± 2.92	0.93	71.00 ± 2.91	1.00
bs	65.12 ± 0.60	76.80 ± 1.36	1.18	65.12 ± 0.60	1.00	69.28 ± 3.00	1.06	77.92 ± 1.18	1.20
dna	93.72 ± 0.72	93.70 ± 0.91	1.00	92.72 ± 0.91	0.99	93.44 ± 0.58	1.00	93.75 ± 0.56	1.00
sp	93.95 ± 0.34	93.92 ± 0.34	1.00	94.01 ± 0.06	1.00	94.20 ± 0.55	1.00	93.73 ± 0.24	1.00
car	90.34 ± 0.31	97.05 ± 0.49	1.07	95.78 ± 0.49	1.06	90.05 ± 2.65	1.00	95.25 ± 0.40	1.05
nur	96.45 ± 0.18	99.41 ± 0.05	1.03	99.54 ± 0.07	1.03	90.17 ± 2.13	0.93	98.07 ± 0.10	1.02
soy	88.28 ± 1.98	89.17 ± 1.17	1.01	89.46 ± 1.55	1.01	82.13 ± 2.50	0.93	89.16 ± 1.39	1.01
ave	84.19	85.05	1.01	84.67	1.00	81.19	0.97	85.27	1.02
	C4.5	Global							
	k-way	BU-IG-G		BU-CC-G		TD-IG-G		TD-CC-G	
	accuracy	accuracy	ratio	accuracy	ratio	accuracy	ratio	accuracy	ratio
ad	86.02 ± 0.33	81.64 ± 0.81	0.95	86.08 ± 0.32	1.00	84.08 ± 0.28	0.98	85.32 ± 0.30	0.99
b5	83.51 ± 0.31	83.02 ± 0.30	0.99	82.37 ± 0.36	0.99	77.60 ± 1.76	0.93	83.42 ± 0.22	1.00
b8	85.06 ± 0.38	68.50 ± 4.28	0.81	83.60 ± 0.69	0.98	79.60 ± 0.42	0.94	84.68 ± 0.76	1.00
sm	72.60 ± 0.47	72.31 ± 0.17	1.00	71.96 ± 0.44	0.99	69.93 ± 0.66	0.96	73.89 ± 0.36	1.02
bc	71.00 ± 2.25	72.04 ± 1.76	1.01	71.34 ± 2.65	1.00	72.75 ± 2.43	1.02	70.29 ± 1.79	0.99
bs	65.12 ± 0.60	67.36 ± 1.88	1.03	68.32 ± 1.80	1.05	74.24 ± 2.01	1.14	77.76 ± 2.11	1.19
dna	93.72 ± 0.72	94.22 ± 0.55	1.01	94.38 ± 0.36	1.01	93.88 ± 0.58	1.00	94.54 ± 0.60	1.01
sp	93.95 ± 0.34	93.92 ± 0.34	1.00	94.04 ± 0.37	1.00	93.67 ± 0.51	1.00	94.80 ± 0.35	1.01
car	90.34 ± 0.31	85.82 ± 0.42	0.95	94.44 ± 0.77	1.05	87.10 ± 1.71	0.96	92.42 ± 0.89	1.02
nur	96.45 ± 0.18	86.92 ± 0.17	0.90	97.64 ± 0.12	1.01	92.32 ± 0.30	0.96	97.90 ± 0.12	1.02
soy	88.28 ± 1.98	89.90 ± 1.42	1.02	88.28 ± 1.98	1.00	88.72 ± 1.90	1.00	88.87 ± 2.06	1.01
ave	84.19	81.42	0.97	84.77	1.01	83.08	0.99	85.81	1.02

Table 2. Classification accuracies of decision trees produced using alternative methods for grouping attribute values. The columns labelled *accuracy* are the mean and standard deviation for 5 cross-validation runs. The columns labelled *ratio* are the ratio of the respective accuracies over the control, $k - way$ method.

preclude the inclusion of timing data, but as might be expected, the local methods were generally slower.

[1] BHPS data may be obtained from Data Archive, University of Essex, Colchester, CO4 3SQ, UK.

5 Discussion

As can be seen, the four local methods were not consistently more accurate than the control, $k - way$, but three of the four global methods were generally less accurate. The exception was TD-CC-G whose accuracy was very close to that of the control for all data sets except *bs* for which it performed significantly better. On the other hand the global methods usually produced considerably smaller trees than either local methods or the control. Indeed in three of the four cases they appeared to be grouping too much with a consequent loss of accuracy.

	C4.5	Local							
	k-way	BU-IG-L		BU-CC-L		TD-IG-L		TD-CC-L	
	nodes	nodes	ratio	nodes	ratio	nodes	ratio	nodes	ratio
ad	548 ± 49	706 ± 12	1.29	553 ± 20	1.01	112 ± 22	0.20	99 ± 17	0.18
b5	1051 ± 49	1204 ± 48	1.15	1133 ± 137	1.08	949 ± 60	0.90	849 ± 33	0.81
b8	789 ± 69	1144 ± 29	1.45	1123 ± 20	1.42	542 ± 270	0.69	850 ± 50	1.08
sm	1790 ± 73	2294 ± 67	1.28	1968 ± 199	1.10	676 ± 115	0.38	996 ± 25	0.56
bc	11 ± 4	29 ± 5	2.64	22 ± 3	2.00	27 ± 5	2.45	24 ± 2	2.18
bs	35 ± 2	63 ± 4	1.80	83 ± 8	2.37	55 ± 7	1.57	63 ± 4	1.80
dna	162 ± 4	121 ± 1	0.75	153 ± 4	0.94	145 ± 9	0.90	139 ± 6	0.86
sp	314 ± 5	170 ± 4	0.54	189 ± 10	0.60	152 ± 3	0.48	118 ± 3	0.38
car	152 ± 2	103 ± 3	0.68	103 ± 1	0.68	84 ± 5	0.55	70 ± 2	0.46
nur	466 ± 11	310 ± 4	0.67	291 ± 1	0.62	241 ± 17	0.52	207 ± 3	0.44
soy	86 ± 1	89 ± 4	1.03	98 ± 3	1.14	90 ± 3	1.05	81 ± 1	0.94
ave	491	567	1.21	520	1.18	279	0.88	318	0.88
	C4.5	Global							
	k-way	BU-IG-G		BU-CC-G		TD-IG-G		TD-CC-G	
	nodes	nodes	ratio	nodes	ratio	nodes	ratio	nodes	ratio
ad	548 ± 49	25 ± 1	0.05	238 ± 3	0.43	33 ± 2	0.06	100 ± 4	0.18
b5	1051 ± 49	259 ± 10	0.25	1180 ± 33	1.12	896 ± 31	0.85	978 ± 55	0.93
b8	789 ± 69	190 ± 13	0.24	943 ± 6	1.20	966 ± 70	1.22	691 ± 27	0.88
sm	1790 ± 73	1067 ± 59	0.60	1663 ± 123	0.93	2090 ± 63	1.17	167 ± 27	0.09
bc	11 ± 4	10 ± 2	0.91	10 ± 2	0.91	6 ± 0	0.55	6 ± 0	0.55
bs	35 ± 2	12 ± 0	0.34	47 ± 1	1.34	14 ± 1	0.40	36 ± 5	1.03
dna	162 ± 4	114 ± 3	0.70	134 ± 2	0.83	121 ± 2	0.75	117 ± 12	0.72
sp	314 ± 5	170 ± 4	0.54	183 ± 3	0.58	143 ± 3	0.46	140 ± 5	0.45
car	152 ± 2	23 ± 1	0.15	56 ± 1	0.37	30 ± 2	0.20	38 ± 3	0.25
nur	466 ± 11	30 ± 1	0.06	318 ± 15	0.68	80 ± 2	0.17	249 ± 12	0.53
soy	86 ± 1	69 ± 1	0.80	86 ± 1	1.00	91 ± 1	1.06	89 ± 1	1.03
ave	491	179	0.42	442	0.85	406	0.63	237	0.60

Table 3. Number of nodes in classification trees produced using alternative methods for grouping attribute values. The columns labelled *nodes* indicate the mean and standard deviation for 5 cross-validation runs. The columns labelled *ratio* indicate the ratio of the respective number of nodes over the control, $k - way$ method.

Again the exception was TD-CC-G, which was, rather surprisingly, not only more accurate but also more consistent in producing small trees. In some cases the local methods actually produced trees that were significantly larger than those generated by the control method.

The story concerning bottom-up and top-down methods is more muddled. Similarly neither Information Gain Ratio nor χ^2 plus Cramer's V showed a consistent advantage over the other.

Nevertheless it is possible to draw at least one firm and useful conclusion from these results. Of all those tested, TD-CC-G would appear to be the best method for reducing

fragmentation in decision tree construction. It achieves the higher-end of the predictive accuracy compared while producing markedly smaller tree sizes. Fortunately it also required less computation time than any other method for all but one of the data sets.

Acknowledgements

We are grateful to the ESRC's programme on the ALCD for supporting part of the work reported in this paper under grant number H519255030.

References

1. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, Pacific Grove, CA., 1984.
2. L. A. Breslow and D. W. Aha. Simplifying Decision Trees: A Survey. *Knowledge Engineering Review*, 12:1–40, 1997.
3. B. Cestnik, I. Konoenko, and I. Bratko. A knowledge elicitation tool for sophisticated users. In I. Bratko and N. Lavrac, editors, *Progress in Machine Learning.* Sigma Press, Wilmslow, England, 1987.
4. P. R. Cohen and D. Jensen. Overfitting Explained. In *Proc. Sixth International Workshop on Artificial Intelligence and Statistics*, pages 115–122, FL, 1997. Ft. Lauderdale.
5. B. S. Everitt. *Cluster Analysis.* Heinemann, London, 2nd edition, 1980.
6. U. M. Fayyad and K. B. Irani. The attribute selection problem in decision tree generation. In *Proc. Tenth National Conference on Artificial Intelligence*, pages 104–110, San Jose, CA., 1992. AAAI Press.
7. J. Healey. *Statistics: A Tool For Social Research.* Wadsworth, Belmont, CA., 1990.
8. E. Hunt, J. Martin, and P. Stone. *Experiments in Induction.* Academic Press, New York, 1966.
9. R. Kohavi, G. John, D. Manley, and K. Pfleger. MLC++: Amachine learning library in C++. In *Tools with Artificial Intelligence*, pages 740–743. IEEE Computer Society Press, 1994.
10. I. Kononenko. A counter example to the stronger version of the binary tree hypothesis. In *ECML-95 Workshop on Statistics and Machine Learning in KDD*, Crete, 1995.
11. T. Oates and D. Jensen. The Effects of Training Set Size on Decision Tree Complexity. In *The Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 379–390, 1997.
12. J. R. Quinlan. The effect of noise on concept learning. In R.S.Michalski, J.G.Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach. Volume II.* Morgan Kaufman Publ. Inc., Los Altos, CA, 1986.
13. J. R. Quinlan. *Programs for Machine Learning.* Morgan Kaufman Publ. Inc., Los Altos, CA, 1993.
14. J. R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.