Value Grouping for Binary Decision Trees

Neil C. Berkman

Computer Science Department
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003-4601

CMPSCI Technical Report 95-19

April 8, 1995

Contents

Value Grouping for Binary Decision Trees

Neil C. Berkman [*]
berkman@cs.umass.edu
University of Massachusetts at Amherst
Computer Science Department
Amherst, MA 01003-4601

Abstract

The tests associated with the internal nodes of a binary decision tree can take on a variety of forms; for instance, they can test whether a given attribute has a given value, or more generally, whether it has a value in a given set. This report examines whether allowing the latter form of test (which we call a value group) provides significant advantages over allowing only single-valued tests. The results of an empirical study comparing performance of these two approaches on a variety of learning tasks are reported. For the data sets studied, value grouping is shown to have no strong effect on classification accuracy, although it does result in smaller trees.

1 Introduction

An issue that has not been explored very deeply in the decision tree induction literature is the effect of allowing a group of values to be used as a test rather than a single value. Extending the range of tests in this way has potential as a simple form of *constructive induction* and may aid in solving the *replication problem.* Although single-value tests are the norm, value grouping is available as an option in some of best known decision tree induction systems (Quinlan, 1993; Breiman, Friedman, Olshen & Stone, 1984). This project is an attempt to assess the effects of value grouping through an empirical study. The focus is on value grouping methods designed for use with algorithms producing binary trees.

[*]This material is based upon work supported by the National Science Foundation under Grant No. IRI-9222766 and by a grant from the Digital Equipment Corporation.

2 Decision Tree Induction

The objective of decision tree induction is to produce a representation of a concept to be used to classify objects. This is accomplished by generalizing from a set of training instances, where each instance is a class label paired with a vector of values for a predefined set of attributes. Decision tree induction typically proceeds by partitioning a set of training examples according to some test. This process is performed recursively until all examples in a partition are of the same class. The result is a tree in which each internal node represents a test, each branch represents a value of a test, and each leaf corresponds to a set of examples of the same class, or for which no more tests are informative. Each test generally corresponds to an attribute. For a discrete attribute, a branch may represent either a single value or a group of values. Continuous attributes are handled differently; one common method is to a construct binary test using a cutpoint, with all values below the cutpoint sent to one branch and all values above it sent to another. Another method is to discretize continuous attributes and treat the intervals as discrete attributes.

Some algorithms produce only binary trees. A test may take one of a variety of forms, such as "attribute = value," "attribute $\in \{\text{value}_1, \ldots, \text{value}_k\}$," or "attribute $<$ cutpoint," where the two branches associated with the test node represent "true" and "false." As there is strong evidence that algorithms producing trees with only binary tests tend to perform better than those allowing multi-valued tests (Fayyad, 1991), this paper restricts its attention to binary trees.

3 Rationale for Value Grouping

Intuitively, grouping is desirable because sometimes values of an attribute naturally "belong together." For example, for an attribute taking a letter as a value, in some situations it might be advantageous to treat the set of vowels as a group. As such, one may think of value grouping as a limited form of *constructive induction*, in which primitive features (such as letters) are automatically combined in some way to produce higher-level features (such as "vowel"). A potential advantage of this ability to construct high-level features would be to reduce the need for a human domain expert to construct such features before presenting data to a learning algorithm.

Value grouping may also help alleviate the *replication problem* (Pagallo, 1989). This problem arises when the tests available to a decision tree algorithm are insufficient to allow a concept to be represented without some duplicated subtrees. The problem is not just that the tree produced will be larger than necessary; replication causes the training set to become overly partitioned. As a result, tests are selected on the basis of less data, increasing the probability that tests of lower quality will be selected. Thus, replication may cause classification accuracy to suffer.

As an illustration of this, consider a set of instances of the form $\langle A, B, \text{class}\rangle$, where $A \in \{a1, a2, a3, a4, a5, a6\}$, $B \in \{b1, b2, b3, b4, b5, b6\}$, and class $\in \{+, -\}$. The class label of an instance is determined by the expression:

$$A \in \{a1, a2, a3, a4\} \wedge B \in \{b1, b2\} \tag{1}$$

An instance satisfying this expression is labeled with class "+," otherwise it is labeled "−." If only single-valued tests are permitted, the smallest tree that can be generated for this concept will contain thirteen nodes. One such minimum-sized tree is shown in Figure 1, using the convention that the left branch is the "true" branch of a node and the right branch is the "false" branch. Note that this tree contains a replicated subtree. If value grouping is permitted, the concept can be represented by a five-node tree, as shown in see Figure 2.

4 Tree Replication Example

As a concrete illustration of the effects of the replication problem I present the following comparison of trees produced with and without value grouping for the concept described above.

4.1 Method

Training sets of size ten to 150 (and all multiples of ten in between) were used, and ten sets of each size were created. Each instance was generated by randomly selecting values for the attributes A and B according to a uniform distribution and labeling the resulting instance according to the expression given in (1). The ITI decision tree induction system (Utgoff, 1994) was used to create trees for each of the training sets. Two versions were run; the *single*

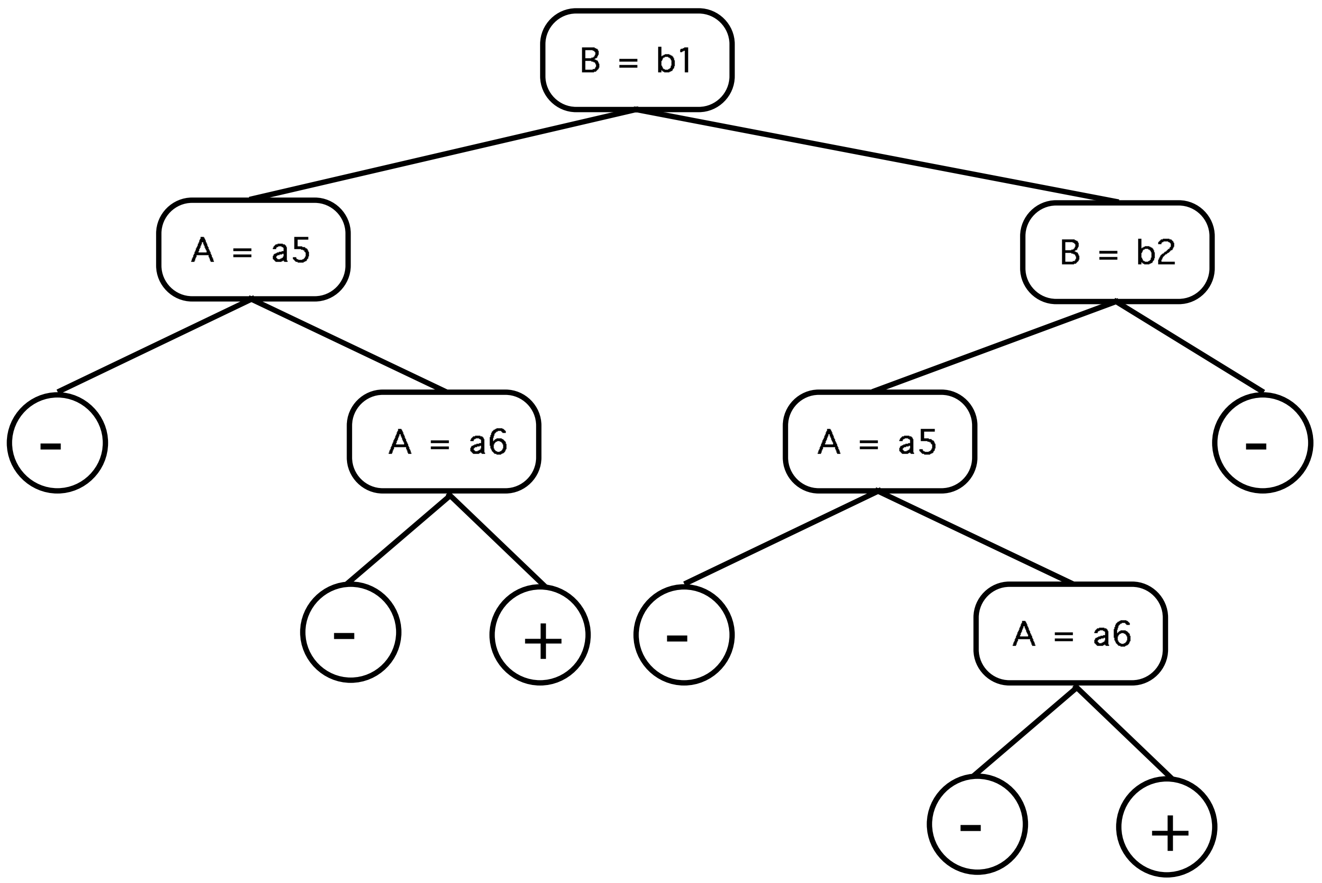

Figure 1: Minimum-sized tree (no value grouping)

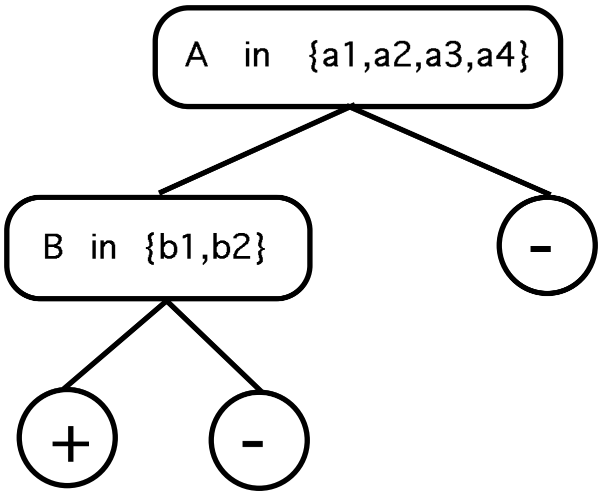

Figure 2: Minimum-sized tree (value grouping permitted)

value version considers only tests of single values whereas the *value grouping* version considers as tests all subsets of values for each attribute. Each selects a test with maximum gain ratio (Quinlan, 1993), a commonly used information-theoretic metric. Decision trees were created for each training set using both methods. Classification accuracy was measured on a test set containing the 36 possible instances, each represented once. Given that every possible instance was tested, a tree scoring 100% accuracy can be said to be a "correct" tree.

4.2 Effects On Size & Accuracy

Figure 3 shows the average size of trees created by each method for the various sizes of training set. Note that, although trees of at least thirteen nodes are needed to represent the concept correctly using single-valued tests, the average size of trees built by the single value version is below thirteen for all training set sizes below 110 instances. The value grouping method constructs trees large enough to represent the concept (five nodes or more) beginning with training sets of size thirty. Figure 4 shows that the value grouping method does indeed find more accurate trees for training sets of all sizes up to 150, at which point both methods produced correct trees for all ten data sets.

These results indicate that there are situations in which value grouping

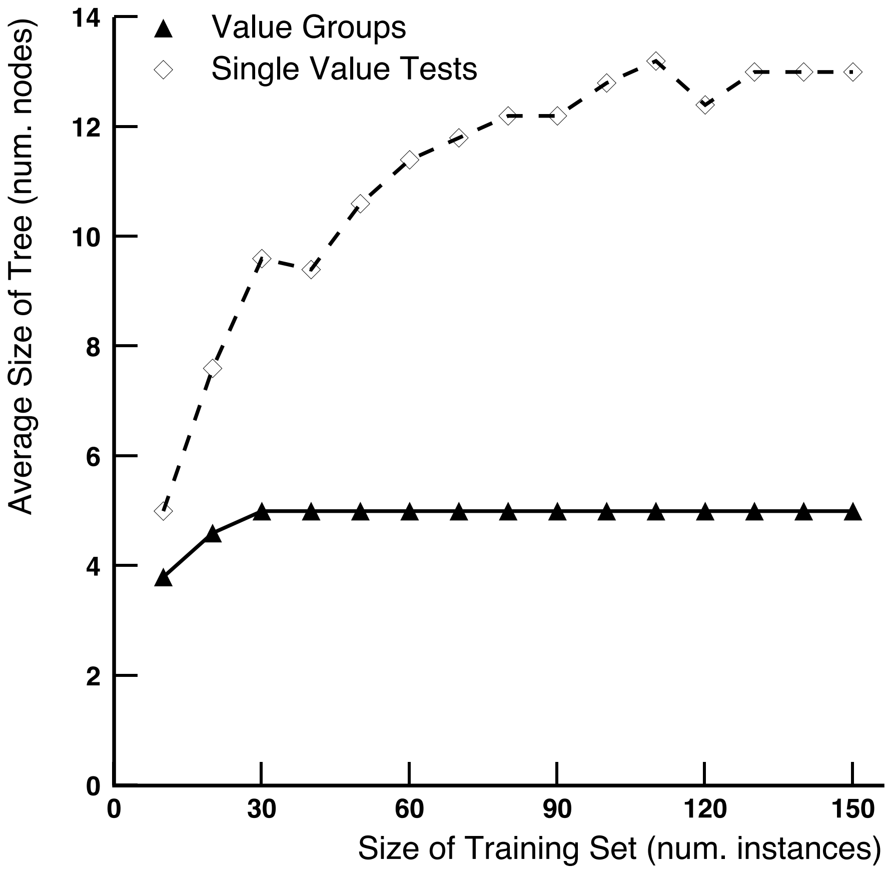

Figure 3: Replication experiment - size results

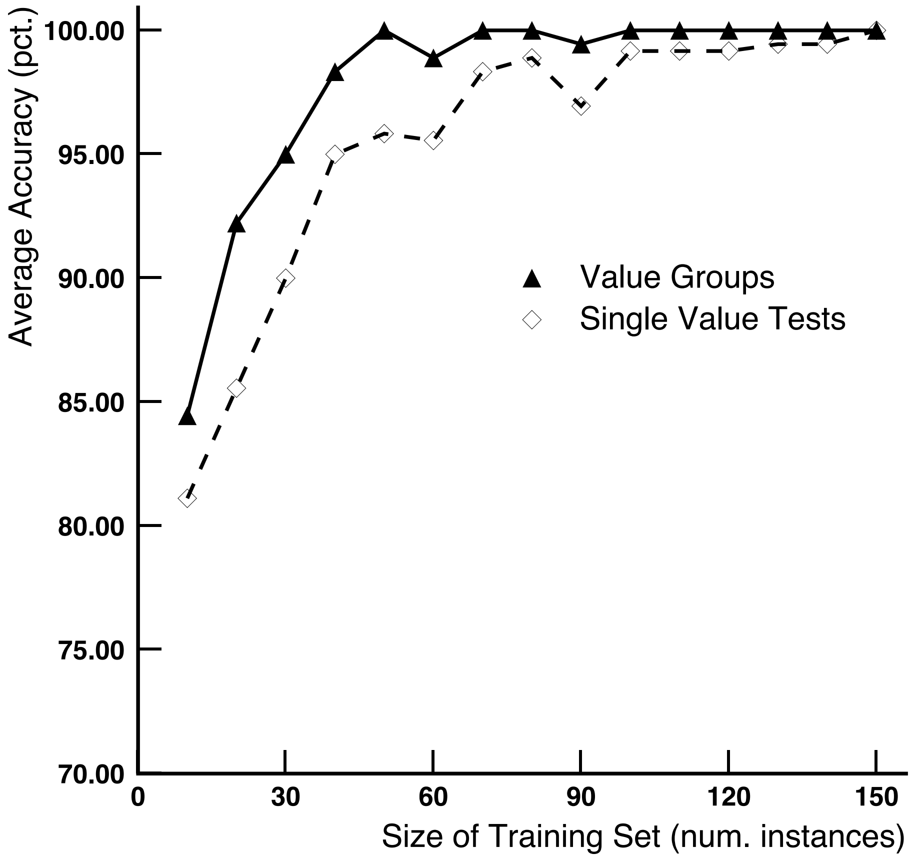

Figure 4: Replication experiment - accuracy results

can improve classification accuracy by solving the tree replication problem. One aim of this project is to determine whether value grouping will have this effect for "real-world" data.

5 Approaches to Value Grouping

Given that value grouping has some potential benefits, let us consider how it can be accomplished. One possibility would be to select a test in the same way in which single-value tests are typically selected - by picking the test with highest gain ratio over all possible tests. This was the method used in the preceding section. The problem with this approach is that there are $2^{N-1} - 1$ possible value groups for an attribute of N values, and thus for large N considering all groups would be computationally infeasible.

One alternative would be to use a polynomial-time heuristic to find a candidate test for each attribute and then to select the candidate with highest gain ratio to be used in the tree. Given that the tests produced by the heuristic will ultimately be evaluated by gain ratio, the goal of the heuristic should be to produce a test with high gain ratio.

This paper will evaluate a heuristic similar to that used in the ASSISTANT-86 system (Cestnik, Kononenko & Bratko, 1987). It is based upon the idea of *sequential forward selection*, and it selects a test for a given attribute as follows. First, all groups containing one value are considered. Of these, the test with highest gain ratio becomes the "current best." Next, all tests that can be created by adding a second value to the group are considered. If the maximal gain ratio over these tests is greater than that of the current best, the test achieving this becomes the new current best. The process is repeated until no improvement is made over the current best, at which point the current best is selected as the test for the attribute.

This heuristic will always pick a test with gain ratio at least as high as the best single-valued test. This does not, however, guarantee that trees produced using the heuristic will have greater classification accuracy than those with only single-valued tests. Gain ratio, like all test-selection metrics, is a heuristic, and tests with high gain ratio are not guaranteed to yield more accurate trees than those with lower gain ratio.

6 Empirical Evaluation

In order to assess the effects of value grouping, I undertook an empirical comparison of three methods of test selection - two value grouping and one non-value grouping.

6.1 Methods Tested

The three methods tested were:

- Exhaustive (EXH)

  This method considers all possible value groups for each attribute and finds a group with maximal gain ratio.

- Sequential Forward Selection (SFS)

  This is the heuristic described in the preceding section.

- Single Value (SNG)

  This method considers the N possible single-valued tests for each attribute and finds a test with maximal gain ratio from among these.

The implementations of these differ only in the method used to select a test for an attribute. Other features shared by all three implementations are as follows:

- All were implemented using the ITI incremental decision tree induction system, which produces binary trees. Batch mode was used for all experiments.
- Numeric attributes were discretized using the *multiple interval discretization* method (Fayyad, 1991). The resulting intervals were treated in the same manner as values of a discrete attribute.
- One test was selected for each attribute (using either the EXH, SNG, or SFS method). Of these, the test with maximum gain ratio among those with at least average gain was selected to be used in the tree. When more than one test had the maximum gain ratio an arbitrary test was chosen. The tie-breaking method was the same for all three methods.

6.2 Questions

This experiment attempts to answer the following questions:

- How do the trees produced by the three methods compare with respect to classification accuracy?

  This question is asked because classification accuracy is the most direct measure of the quality of a decision tree.

- How do the trees produced by the three methods compare with respect to size?

  If value grouping produces larger trees, it is not solving the tree replication problem.

- How often is the test with maximal-gain ratio a group of more than one value?

  Another way to phrase this question is "how often is the test selected by EXH different from that selected by SNG?"

  If the answer to this question is "almost always," it may not be worth the extra computational expense involved in considering value groups.

- How do the tests selected by the sequential forward selection heuristic compare to those selected by exhaustive search?

  This asks how well the heuristic approximates exhaustive search.

Note that the first two questions ask about the *trees* produced by the various methods while the last two concern the *tests* selected. It is certainly important to look at the quality of the trees produced; however, classification accuracy is also affected by the test selection metric used (in this case, gain ratio). To eliminate the variance due to the heuristic nature of gain ratio, it will also be informative to examine directly the gain ratios of tests selected by the various methods. Specifically, we can perform exhaustive search to obtain a complete ordering of all possible tests with respect to gain ratio. We can then calculate the rankings of the tests found by SFS and SNG.

6.3 Experimental Method

For this experiment, performance of the various methods was measured over a number of different learning tasks. The process by which these tasks were chosen is described in the next section. Ten four-fold cross-validation runs were performed for each task using each method. For each cross-validation run, the data set was split into four equal-sized parts. Each of these parts was used as a test set for one run, with the remainder of the data used as a training set. Appendix A outlines the algorithm used to build and gather statistics on a decision tree.

One set of statistics was computed to answer the first two questions listed above, which concern the trees produced by the three methods. These statistics are the average size of the tree produced and the average classification accuracy on the test set, recorded for each run using each method and averaged over the four folds. The algorithm for computing these is given in Appendix A, Procedure GET_TREE_STATISTICS.

Another set of statistics was computed to answer the last two questions, which concern the tests chosen by the three methods. These were gathered only on the runs using the EXH method, since computing them required ranking all possible tests for each attribute by gain ratio and determining where the tests found by the SNG and SFS methods ranked. It is important to note that for attributes of three or fewer values, the SNG method will consider all possible tests and thus will produce the same test as the EXH method. The SFS method will consider all possible tests for attributes of four or fewer values. Statistics were gathered to compare the tests found by SFS and SNG in only those situations in which the attribute under consideration had a sufficient number of values so that the behavior of these methods differed from that of EXH. The method for computing these statistics is described in Appendix A, Procedure GET_TREE_STATISTICS.

6.4 Learning Tasks

Data sets representing eleven different learning tasks were used in this comparison. Some of these have only discrete attributes, some only continuous, and some have both. No data set having only discrete attributes with fewer than four values was used, since all three methods will consider all possible tests for an attribute with three or fewer values, and will thus produce identical trees. Similarly, all three methods will produce identical trees for

		Discrete Attributes		Continuous Attributes	
Data Set	Instances	Number	Max Vals	Number	Max Intvls
canasta	2299	14	1210	1	19
crx	690	6	14	9	3
glass-no-id	214	0	N/A	8	6
nettalk	5438	7	27	0	N/A
landsat	1000	0	N/A	7	6
plural	1937	15	27	0	N/A
promoter	106	57	4	0	N/A
road	2056	0	N/A	6	7
soybean	683	35	7	0	N/A
splice	3190	60	6	0	N/A
vowel	528	0	N/A	10	8

Table 1: Data Set Characteristics

continuous attributes with three or fewer values. Since the number of intervals produced for a continuous attribute is not determined until run time, a preliminary investigation was undertaken to find data sets for which the discretization procedure would produce four or more intervals for at least some attribute.

Table 1 summarizes some characteristics of the data sets used. The column labeled "Max Vals" gives the maximum number of values over the discrete attributes. "Max Intvls" gives the maximum number of intervals produced by the intervalization procedure. Since this occurs at run time, the figures given are based on runs using all of the data set for training and SNG as the test selection method. Note that the data sets used all have either a "Max Vals" or a "Max Intvls" of four or greater.

Nine of the learning tasks used were drawn from the repository kept at the University of California at Irvine. Only five tasks in the entire repository contain at least one discrete attribute with four or more values, and all of these were used in this investigation. Fourteen data sets with only continuous attributes were selected aribtrarily from the repository, and the four used in the experiment were the only ones of these that met the criterion listed above.

	Accuracy (pct.)		
Data Set	EXH	SFS	SNG
canasta	N/A	81.33	86.14
crx	80.46	80.49	81.17
glass-no-id	64.58	64.68	65.79
nettalk	N/A	82.29	82.67
landsat	80.50	80.50	80.50
plural	N/A	95.20	95.11
promoter	69.54	69.54	66.57
road	76.82	76.82	77.00
soybean	86.29	86.33	86.50
splice	90.59	90.59	89.84
vowel	72.12	72.31	72.84

Table 2: Accuracy Results

6.5 Results

The questions posed in Section 6.2 can now be answered:

6.5.1 How do the trees produced by the three methods compare with respect to classification accuracy?

Table 2 shows the average percentage of the training set correctly classified for each method on the data sets used. Note that the exhaustive method was not run for the canasta, nettalk and plural data sets. This is because these data sets contain attributes with large numbers of values, making the exhaustive method computationally infeasible.

Paired t-tests were run on each pair of methods to determine statistical significance of these results. In no case were the differences between the EXH and SFS methods significant. For the splice data set, the differences in accuracy between SNG and both EXH and SFS were significant at the .05 level. For the nettalk and canasta data, the difference between SNG and SFS was significant at the .05 level. In only one of these cases (splice) were the trees built with the value grouping methods more accurate on average than those built with the single value method.

	Size (num. nodes)		
Data Set	EXH	SFS	SNG
canasta	N/A	292.65	495.40
crx	118.80	119.10	127.90
glass-no-id	58.60	58.45	59.90
nettalk	N/A	997.75	1436.70
landsat	193.90	193.90	192.35
plural	N/A	127.25	172.55
promoter	23.55	23.55	27.15
road	406.10	406.15	411.25
soybean	140.15	140.15	151.40
splice	279.70	279.70	322.55
vowel	124.30	124.10	127.65

Table 3: Size Results

6.5.2 How do the trees produced by the three methods compare with respect to size?

Table 3 shows average tree size in number of leaves. As with classification accuracy, in no case were the differences in size significant between EXH and SFS. For the following data sets, the differences in tree size between SNG and both EXH and SFS were significant at the .05 level: crx, glass-no-id, promoter, road, soybean, splice, vowel. For canasta, nettalk, and plural, the differences between SNG and SFS were significant at the .05 level. In all of these cases, the trees built with value grouping were smaller on average than those built with the single value method.

One conclusion that can be drawn from these results is that value grouping did cause significantly smaller trees to be built. Smaller trees are generally considered to be more understandable, so by this criterion, allowing value groups provides an advantage over single-valued tests.

6.5.3 How often is the test with maximal-gain ratio a group of more than one value?

Table 4 summarizes the average rankings by gain ratio of tests found by SNG. As described in Section 6.3, these statistics were gathered by comparing tests

Data Set	Total Tests	Percent Matches (Total)	Tests w/ > 3 Vals	Percent Matches (> 3 Vals)	Percentile Rank of Non-Matches
crx	735.00	93.68	69.03	32.67	89.17
glass-no-id	249.37	98.37	5.68	28.19	85.41
landsat	652.08	98.78	9.93	19.65	83.61
promoter	630.33	71.63	477.98	62.59	85.71
road	1412.25	98.76	21.50	18.37	84.77
soybean	1090.00	93.47	149.73	52.48	84.35
splice	7789.37	69.39	5734.67	58.43	85.77
vowel	616.47	94.85	46.42	31.61	80.37
plural	632.00	61.23	431.00	43.16	91.56

Table 4: Gain Ratio Ranking Results: SNG

chosen by SNG to those chosen by EXH. Given the large numbers of values per attribute for the canasta and nettalk data, EXH could not be run for these. It is unfortunate that runs could not be performed for data sets with many values per attribute, since it seems likely that the differences between the non-exhaustive and exhaustive methods would be greater for these. To provide some clue as to whether this is the case, one run (that is, one fold of one cross-validation) was performed for the plural data set [1]. For all other data sets, the figures presented are averages over all runs using EXH.

Table 4 gives the statistics for the SNG method computed in Procedure GET_TREE_STATISTICS (see Appendix A). Briefly, a "match" on a given attribute indicates that the test chosen by the SNG method matched that found by EXH; in other words, that value grouping was unnecessary to find the test with maximal gain ratio for that attribute. Separate statistics were computed for those situations in which the attribute under consideration had more than three values, as SNG will perform the same as EXH for attributes of three or fewer values. The "Percentile Rank of Non-Matches" indicates the average the percentile ranking of the test chosen by SNG for those cases when the chosen test has less than maximal gain ratio. This is intended to indicate the quality of tests selected when SNG fails to select the "best" test.

[1]This one run took over two days to complete on a DEC Alpha

Data Set	Total Tests	Percent Matches (Total)	Tests w/ > 4 Vals	Percent Matches (> 4 Vals)	Percentile Rank of Non-Matches
crx	735.00	98.93	53.67	85.28	97.16
glass-no-id	249.37	99.97	1.58	95.24	94.55
landsat	652.08	99.99	5.65	99.12	93.44
promoter	630.33	100.00	0.00	N/A	N/A
road	1412.25	100.00	5.47	99.09	90.00
soybean	1090.00	99.88	26.20	94.94	95.06
splice	7789.37	99.97	280.37	99.10	92.44
vowel	616.47	99.87	16.67	95.20	94.18
plural	632.00	90.82	370.00	84.32	98.47

Table 5: Gain Ratio Ranking Results: SFS

It is apparent from the low figures in the "Percent Matches $>$ 3 Vals)" column that the test with maximal gain ratio is often a group of more than one value. This indicates that if the goal is to find the test with highest gain ratio, it is important to allow value groups.

6.5.4 How do the tests selected by the sequential forward selection heuristic compare to those selected by exhaustive search?

Table 5 summarizes the average rankings by gain ratio of tests found by SFS. The columns of this table have the same meanings as their counterparts in Table 4. The only difference is that tests with more than four, rather than three, values are considered separately, since the SFS heuristic will consider all possible groups for attributes with four or fewer values.

These results seem to indicate that for these data sets, the heuristic provides a good approximation to exhaustive search. For all but two data sets, 95% or more of the tests found by SFS had the highest possible gain ratio. Also, for all data sets, tests found by SFS with less than maximal gain ratio averaged in the 90th percentile or higher. This indicates that when SFS does not find the highest gain ratio test, it tends to find tests with gain ratios that rank highly compared to all possible tests. By this measure, SFS can be considered a good approximation to exhaustive search.

7 Conclusions

Given the evidence that accuracy gains should result from a reduction in tree replication, the fact that value grouping had no strong effect on accuracy in these experiments indicates that it did not solve the replication problem. However, it is possible that replication is simply not a problem for any of the data sets tested. The fact that value grouping produced smaller trees with no significant change in accuracy may be sufficient justification for its use, particularly in situations where understandability is important. Furthermore, it is clear that a computationally feasible heuristic exists that closely approximates the performance of exhaustive search in terms of the gain ratio of tests chosen. A surprising aspect of these results is that value grouping frequently produced tests with higher gain ratio than any single-valued test, but this did not lead lead to significantly more accurate trees. Further research, perhaps using attribute selection metrics other than gain ratio, would be helpful in resolving this puzzle.

Appendix A: The Decision Tree Induction Algorithm

Procedure BUILD_TREE(TRAINING_SET,TEST_SET,ATTRIBUTES)

1. (EXH method only) Initialize TOTAL_TESTS, OVER_3_VALS, OVER_4_VALS, SFS_MATCHES_TOTAL, SNG_MATCHES_TOTAL, SNG_MATCHES_OVER_3_VALS, SFS_MATCHES_OVER_4_VALS, SNG_TOTAL_PCTL, and SFS_TOTAL_PCTL to 0.
2. Call BUILD_NODE(INSTANCES,ATTRIBUTES) and set TREE to its return value.
3. Call COMPUTE_TREE_STATS(TREE)
4. (EXH method only) Call COMPUTE_TEST_STATS(TREE)
5. Return TREE

Procedure BUILD_NODE(INSTANCES,ATTRIBUTES)

1. If all instances are of the same class, return leaf node labeled with that class.

```
2. For i = 1 to $N$, the number of ATTRIBUTES.
    (a) Call PICK_TEST_FOR_ATTRIBUTE_{SNG,SFS,EXH} (ATTRIBUTES$_i$) and set TEST$_i$ to its return value.
    (b) Let GAIN$_i$ be the information gain of TEST$_i$
    (c) Let GAIN_RATIO$_i$ be the gain ratio of TEST$_i$
3. If no test has GAIN $> 0$, return leaf node labelled with the majority class over the instances.
4. Let AVG_GAIN_RATIO be the average gain ratio over the TEST$_i$'s
5. Let SELECTED_TEST be the test with highest GAIN_RATIO from among those TEST$_i$'s with GAIN greater than or equal to AVG_GAIN_RATIO.
6. Partition the data according to SELECTED_TEST. Let INSTANCES$_T$ be the set of instances for which SELECTED_TEST is true, and let INSTANCES$_F$ be the set of instances for which SELECTED_TEST is false.
7. Create a new node called NODE with SELECTED_TEST as its test.
8. Call BUILD_NODE(INSTANCES$_T$,ATTRIBUTES), and set the left child of NODE to the result.
9. Call BUILD_NODE(INSTANCES$_F$,ATTRIBUTES), and set the right child of NODE to the result.
10. Return NODE.
```

Procedure PICK_TEST_FOR_ATTRIBUTE_SNG(ATTRIBUTE)

```
1. Compute the gain ratio for each single-value test.
2. Return the test with highest gain ratio.
```

Procedure PICK_TEST_FOR_ATTRIBUTE_SFS(ATTRIBUTE)

```
1. Let $N$ be the number of values for the attribute.
2. Let IN_GROUP = {}
3. Let NOT_IN_GROUP = $\{ATTRIBUTE_1, \ldots, ATTRIBUTE_N\}$
```

```
4. Let BEST_GR be 0.
5. Let IMPROVING be TRUE.
6. While IMPROVING and |NOT_IN_GROUP| $> 0$ do
   (a) Let IMPROVING be FALSE.
   (b) Let NEXT_TO_ADD be NIL.
   (c) For i $=$ 1 to |IN_GROUP|.
       i. Let CURRENT_VALUE be the ith element of NOT_IN_GROUP.
       ii. Remove CURRENT_VALUE from NOT_IN_GROUP and add
           it to IN_GROUP.
       iii. Calculate GR, the gain ratio of the test where all instances
            with a value in IN_GROUP are sent down one branch and all
            others sent down the other branch.
       iv. If GR $>$ BEST_GR, let NEXT_TO_ADD be CURRENT_VALUE
           and let BEST_GR be GR.
       v. Remove CURRENT_VALUE from IN_GROUP and re-add it
          to NOT_IN_GROUP.
   (d) If NEXT_TO_ADD is not NIL, set IMPROVING to TRUE and
       remove CURRENT_VALUE from NOT_IN_GROUP and add it to
       IN_GROUP.
7. Return the test specified by IN_GROUP.
```

Procedure PICK_TEST_FOR_ATTRIBUTE_EXH(ATTRIBUTE)

```
1. Increment TOTAL_TESTS by 1.
2. Let $N$ be the number of values for the attribute.
3. If $N > 3$, increment OVER_3_VALS
4. If $N > 4$, increment OVER_4_VALS
5. Calculate the gain ratio for each of the $2^{N-1} - 1$ possible tests. Rank
   the tests numerically in increasing order by gain ratio. Call the test
   with highest gain ratio $T_{EXH}$, as this is the test that EXH will select.
```

6. Call PICK_TEST_FOR_ATTRIBUTE_SNG(ATTRIBUTE) and set T_{SNG} to its return value. Let RANK_{SNG} be the numerical rank of T_{SNG} in the ranking computed above.

7. Call PICK_TEST_FOR_ATTRIBUTE_SFS(ATTRIBUTE) and set T_{SFS} to its return value. Let RANK_{SFS} be the numerical rank of T_{SFS} in the ranking computed above.

8. If T_{SFS} matches T_{EXH}, then increment SFS_MATCHES_TOTAL, and if $N > 4$, increment SFS_MATCHES_OVER_4_VALS.

9. if T_{SFS} does not match T_{EXH}, then let PCTL_{SFS}, the percentile rank of T_{SFS}, be $\frac{\mathrm{RANK}_{SFS}}{2^{N-1}-1} \times 100$. Increment SFS_TOTAL_PCTL by PCTL_{SFS}.

10. If T_{SNG} matches T_{EXH}, then increment SNG_MATCHES_TOTAL, and if $N > 3$, increment SNG_MATCHES_OVER_3_VALS

11. if T_{SNG} does not match T_{EXH}, then let PCTL_{SNG}, the percentile rank of T_{SNG}, be $\frac{\mathrm{RANK}_{SNG}}{2^{N-1}-1} \times 100$. Increment SNG_TOTAL_PCTL by PCTL_{SNG}.

12. Return T_{EXH}.

Procedure COMPUTE_TREE_STATS(TREE,TEST_SET)

1. Classify each instance from TEST_SET using TREE. Let ACCURACY be the percentage of instances correctly classified.

2. Let SIZE be the number of nodes in the tree.

Procedure COMPUTE_TEST_STATS(TREE)

1. Let SNG_PERCENT_MATCHES_TOTAL be $\frac{\text{SNG\_MATCHES\_TOTAL}}{\text{TOTAL\_TESTS}}$.

2. Let SNG_PERCENT_MATCHES_OVER_3_VALS be $\frac{\text{SNG\_MATCHES\_OVER\_3\_VALS}}{\text{OVER\_3\_VALS}}$.

3. Let SNG_NON-MATCHES be TOTAL_TESTS−SNG_MATCHES_TOTAL.

4. Let SNG_NON-MATCH_PCTL_RANK be $\frac{\text{SNG\_TOTAL\_PCTL}}{\text{SNG\_NON-MATCHES}}$.

5. Let SFS_PERCENT_MATCHES_TOTAL be $\frac{\text{SFS\_MATCHES\_TOTAL}}{\text{TOTAL\_TESTS}}$.

6. Let SFS_PERCENT_MATCHES_OVER_4_VALS be $\frac{\text{SFS\_MATCHES\_OVER\_4\_VALS}}{\text{OVER\_4\_VALS}}$.
7. Let SFS_NON-MATCHES be TOTAL_TESTS−SFS_MATCHES_TOTAL.
8. Let SFS_NON-MATCH_PCTL_RANK be $\frac{\text{SFS\_TOTAL\_PCTL}}{\text{SFS\_NON-MATCHES}}$.

Acknowledgments

I thank Paul Utgoff for many helpful comments.

References

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont, CA: Wadsworth International Group.

Cestnik, B., Kononenko, I., & Bratko, I. (1987). ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In Bratko & Lavrac (Eds.), *Progress in Machine Learning.* Wilmslow, UK: Sigma Press.

Fayyad, U. M. (1991). *On the induction of decision trees for multiple concept learning.* Doctoral dissertation, Computer Science and Engineering, University of Michigan.

Pagallo, G. (1989). Learning DNF by decision trees. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 639-644). Detroit, Michigan: Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* Morgan Kaufmann.

Utgoff, P. E. (1994). An improved algorithm for incremental induction of decision trees. *Machine Learning: Proceedings of the Eleventh International Conference* (pp. 318-325). New Brunswick, NJ: Morgan Kaufmann.