A STUDY OF OVERFIT IN DECISION-TREE INDUCTION

Thomas L. Duell and Lawrence B. Holder
Department of Computer Science and Engineering
University of Texas at Arlington
Box 19015, Arlington, TX 76019-0015
Email: {duell, holder}@cse.uta.edu

Abstract

This paper studies the overfit phenomenon in machine learning induction algorithms, specifically decision tree induction. Overfitting the training data is a common problem with inductive learning algorithms. The generally accepted countermeasure to overfitting is pruning. This paper studies decision trees built from several datasets, using Quinlan's C4.5 program. We applied different metrics to these decision trees, changing the order in which nodes are expanded into subtrees and studying the effects on the accuracy and size of these subtrees. Results favor an ordering that prefers children having the most number of examples from their parent's minority class. We then group the datasets according to various features and identify patterns and relationships between these features and the accuracy results.

1 INTRODUCTION

Overfitting the training data is a common problem with inductive learning algorithms. Overfitting may be the result of an algorithm learning the training data or patterns that arise by chance, rather than the algorithm learning the underlying concept. We study the overfit phenomenon in the induction of decision trees. For example, a decision tree that correctly classifies all of the training examples may not be as good a classifier on unseen examples as a simpler tree that does not fit all of the training data. The generally accepted countermeasure to overfitting is pruning. Pruning applies a metric to a decision tree (or subtree), attempting to figure out which branches of the tree to keep and which to reduce to a leaf node for optimal performance when classifying unseen examples. The success of pruning methods is considered domain dependent, as will be shown in section 3. The pruning metric will measure the relationship between a node and its children to determine the cut point, the node which no longer gets expanded.

This paper studies decision trees built from several datasets. We applied different metrics to these decision trees, changing the order in which nodes are expanded into subtrees. We studied several orderings on each dataset, extracting the best accuracy (on the testing set), the average accuracy for its subtrees, the smallest subtree having the best accuracy, and the number of subtrees having the best accuracy. We look at fixed cut points for certain orderings, as well as comparing the accuracy and placement of the optimal trees amongst the orderings.

We also extracted the number of attributes, classes, training and testing examples, the percentage of rows with missing values, the percentage of all attribute values missing, the total number of nodes and the depth of the full tree, and the baseline accuracy from each of these trees. We then group the datasets according to these various features along with our results from the different orderings, expecting to find patterns and relationships between these features and the accuracy results mentioned earlier.

The next section reviews the C4.5 program, and section 3 reviews other work related to overfit in decision tree induction. Section 4 presents our experiments, how we expand the subtrees, the features of decision trees which we studied, the accuracy results, and the conclusions to be drawn from the results. The final section presents conclusions and suggests possible future studies.

2 C4.5

C4.5 [Quinlan, 1993] uses information theory to select the tests for non-leaf nodes. Quinlan gives a for-

Proceedings of the 10th Florida Artificial Intelligence Research Symposium, 1997, by the Florida AI Research Society
0-9620-1739-6/97/342 © 1997 FLAIRS

mula for the amount of information at a node in the decision tree. We can also compute the expected information after splitting the node using a particular attribute. The information gain is then computed as the difference between the expected information of the split and the information at the node. This gain criterion is biased in favor of attributes with many outcomes n, so the gain ratio criterion divides the gain by the information in an arbitrary n-way split of the data.

In order to handle unknown attribute values in generating a decision tree, C4.5 will multiply the information gain by the fraction F of cases whose values are known. When C4.5 is attempting to classify an unseen case and encounters a split attribute whose value is unknown, equal weights are passed down for each possible outcome of that test, and the results are polled with the class having the highest weight being chosen as the predicted class.

C4.5 supports three types of tests for its decision nodes. The simplest type of test is one on a discrete attribute with one outcome for each possible value of the attribute. The second type is a binary test on an attribute with continuous numeric values. C4.5 creates the binary tests by choosing a threshold Z so that the possible binary tests are for $A \leq Z$ or $A > Z$. C4.5 only considers threshold values that appear in the data, giving it a finite number of tests to consider. The third type of test allows subgrouping within the possible outcomes of a discrete variable. This third type may be invoked using a non- default option of C4.5. We did not invoke this option in our experiments.

C4.5 has some built in strategies to overcome overfitting, such as the ability not to build a node when there are not enough training examples to justify one (pre-pruning). C4.5 also has a post-pruning mechanism, called pessimistic error pruning, that estimates the error rate of every subtree, and replaces the subtree with a leaf node if the estimated error of the leaf node is lower. When generating pruned trees, our experiments used the default parameter settings for the pre- and post-pruning mechanisms.

3 PRUNING AND OVERFIT

Mingers [1989b] reviews several split-attribute selection strategies. He concludes that the predictive accuracy of induced decision trees is not sensitive to the chosen split measure, and that selecting attributes entirely randomly produces trees that are as accurate as those produced using a measure. Still, he shows that the gain ratio produces the smallest trees of the tested measures, and randomly selecting attributes produces trees twice as large as those produced with an informed measure. After pruning, Mingers notes, that there is little difference in tree size. Since C4.5 uses the gain ratio criterion for selecting split attributes, we can be confident that it makes good choices in building its decision trees, and thus is a good algorithm to analyze.

Mingers [1989a] reviews several pruning strategies. Mingers concludes that there is a significant interaction between pruning and the domain, but no evidence of an interaction between the type of measure used in tree creation and the pruning method. Mingers explains that "generally, the effects of pruning are very strong, reducing large trees to only a few leaves". Mingers includes an earlier version of Quinlan's pessimistic error pruning and concluded that it is more crude than pruning methods which use (require) a separate test set. Pessimistic pruning was the quickest, but gave bad results on certain datasets.

Schaffer [1993] argues that the very practice of pruning (overfit avoidance) applies a potentially harmful bias towards overly simple trees. Schaffer performs a series of experiments in a controlled domain with five binary attributes and a binary class. Schaffer also controls the noise in the experiment by randomly complementing class values. He uses these experiments to illustrate "the fact that all well known overfitting avoidance techniques are ... a form of bias rather than a statistical improvement in inducing decision trees." Schaffer compares the results from a sophisticated strategy which uses pruning, and a naive strategy which simply uses the fully generated tree. The sophisticated strategy outperforms the naive strategy on the first few simple experiments, but when he introduces the parity problem, the naive strategy consistently outperforms the sophisticated strategy. Schaffer also shows that there are more complex problems than simple problems, and the naive strategy will produce the more accurate trees on these.

One of the advantages of pruning is that it removes branches from the decision tree which may be generated by chance occurrences (noise) in the training data. However, Schaffer displays that for the parity problem, "as error rate increases, the naive and sophisticated strategies disagree more often; and at every level of noise the naive strategy proves superior." The error rate in Schaffer's experiments is class noise, and as the error rate in the training examples increases, so too does the noise in the testing examples. When Schaffer experiments with attribute noise, he displays that the value of overfit avoidance "substantially" increases.

Holder [1995] empirically compares intermediate decision trees (IDTs) with full trees (grown by C4.5) and pruned trees (using C4.5's pruning capabilities.) The initial IDT is the root. More IDTs are developed by selecting a node on an IDT and adding its children to create a new IDT. Holder uses a hill-climbing approach to tune C4.5's pruning parameters to generate optimal full and pruned trees (with respect to accuracy on the test set) over 66 datasets. Holder then generates IDTs from these full trees, called IFTs, and pruned trees, called IPTs. Holder extracts features from the datasets and produces 6 hypotheses about the performance of the IDTs, such as hypothesis 1 which states that "IFT has less error than PT" (pruned trees). Holder gives the evidence supporting these hypotheses, then labels the datasets as positive and negative examples of the hypotheses. This allows Holder to use C4.5 to generate rules to define for which types of datasets IFTs outperform trees pruned using C4.5's pruning capabilities and addressing the domain dependence of pruning metrics observed by Mingers and Schaffer.

4 EXPERIMENTS

4.1 Methodology

The 52 datasets used for these experiments all come from the UC-Irvine machine learning repository. We split the 52 datasets into a training set of 70% and a testing set of 30% for each data set, repeating this practice 5 times for each. From this, we generated 260 unpruned and 260 pruned decision trees using C4.5 and its default parameters. Programs expanded the unpruned decision trees to include the number of examples, class distribution, and error count for each node on the training and test data sets. With these new trees, we applied several node orderings (based on the training data statistics) and immediately produced the test set statistics in one pass (per ordering) without reconsulting the training or test data sets.

Eight different node orderings were used for these experiments. Each ordering "generates" n trees (where n is the number of nodes in the full tree) by starting with the root node, then adding one node (according to the ordering) to generate the next tree. The first and second orderings, DEPTH and BREADTH, are simple depth first and breadth first searches. The other orderings are all best first searches using various measurements. The INFO measurement is information based, similar to the way the decision tree is generated. The IMPROVE measurement judges the best node as that which shows the greatest improvement in accuracy (as compared with that node's parent, on the training set). Two more measurements are accuracy oriented (on the training set), measuring the number of correct classifications by a node and selecting the node with the most (CORRECT) or the node with the fewest (INCORRECT). The last two measurements mimic Quinlan's philosophy for converting a decision tree to rules [Quinlan, 1993], which looks for a default class, then finds branches that classify exceptions and puts them into the rules. Our two measurements judge a node as best when it has the most examples from its parent's minority class(es) and when it has the most examples from the overall (or root node's) minority class(es). We call these two PARENT and OVERALL.

All of the orderings start with the root node as the first selected node. The algorithm that grows the trees will add all of the selected node's children to the list of available nodes. The first child will replace the node being taken off the list and the other children are added to the end. The algorithm then selects the available node with the highest measure, depending on the ordering. When two available nodes have the same maximal score, the first node found (closest to the head of the list) will be used. For classification purposes, when not all of a parent's child nodes are expanded, all examples that do not pass the attribute value tests for the expanded nodes will be classified in the parent's majority class.

4.2 The Results by Ordering

We applied each of the node orderings to the 260 unpruned decision trees. We studied each ordering and its accuracies on the testing examples. We use four measures to explore the node orderings. One measure notes the average accuracy of all the trees (AVG). Another measures the high score from the subtree with the best classification rate (HIGH). We also looked at how many (few) nodes were in the smallest tree with the best score (FEW). Finally, we counted how many trees had the best score (MANY) as a measure of how easy or how elusive it was to find the best tree. These results are displayed for each ordering in Table 1.

For nine of the datasets, all eight node orderings attain the same highest accuracy. When reporting how many datasets a particular ordering has the high score for, we will disregard these datasets. All eight orderings have the same score for the root and full tree. MANY is the only measurement that is not normalized. AVG and HIGH are both measuring the number of correct classifications divided by the number of testing examples. FEW measures the percentage of nodes in the smallest, most accurate tree, versus the number of nodes in the full tree. The count

	AVG	HIGH	FEW	MANY	BestAvg	UniqHigh	WorstAvg	LowHigh
DEPTH	71.23	83.37	64.60	4.53	5	3	20	12
BREADTH	75.26	83.89	53.31	6.72	2	**5**	**0**	**0**
INFO	71.80	83.57	64.05	5.50	1	1	7	5
IMPROVE	75.62	84.00	58.35	6.25	9	**5**	1	0
CORRECT	73.39	84.21	65.04	5.20	2	2	9	**0**
INCORR	74.68	83.82	50.76	5.35	12	**5**	3	2
PARENT	**77.33**	**84.07**	**37.46**	**8.55**	**24**	3	2	**0**
OVERALL	75.39	83.74	53.44	6.58	0	0	11	6

Table 1: Performance of each node ordering (best in boldface).

of datasets for which each ordering has the highest and lowest AVG score is included as the BestAvg and WorstAvg columns in Table 1. The count of datasets for which each ordering has the unique high HIGH score is shown in the UniqHigh column, and the count of datasets where every other ordering finds a higher HIGH score is included as LowHigh.

The PARENT ordering's strategy of seeking diversity early allows it to find more accurate trees with far fewer nodes than any of the other orderings. The PARENT ordering is somewhat modeled after C4.5's method for extracting rules from a decision tree. The rule extraction process defines a default rule classifying the majority of cases (the majority class). Similarly, PARENT delays the expansion of segments of the decision tree where the most examples fall into the majority class. The rule extraction process creates rules to classify exceptions to the default rule, and these are the parts of the tree which PARENT visits first.

The DEPTH first search performs poorly on a large number of datasets (relative to the other orderings), but still manages to outperform all other orderings on a few datasets. The BREADTH first ordering produces better than average classification accuracy, occasionally outperforming all other orderings, but never performs poorest. IMPROVE and INCORRECT also generally outperform other orderings, although INCORRECT has problems with more datasets. Despite its similarity to PARENT, OVERALL performs poorer on most datasets.

4.3 The Results by Feature

We ordered the domains according to several features, splitting the domains into two groups, above and below the average value for that feature, hoping to see some simple relationships between the different measures of the datasets. For example, the number of training examples splits the domains into those with more than 768.46 (the average) and those with less. We will refer to datasets which have an above average value for a particular feature as "LARGE" or "larger", and to datasets with a less than average value as "SMALL" or "smaller". The features we investigate include the number of attributes, classes, examples, the percentage of rows with missing values, the percentage of all attribute values missing, the total number of nodes and the depth of the full tree, and the baseline accuracy of choosing the majority class. We also investigate dividing the datasets based on the AVG, HIGH, FEW and MANY scores from Table 1, and the percentage of examples correctly classified by the C4.5 post-pruned trees.

For example, Table 2 shows the properties of the SMALL and LARGE sets divided based on the average number of attributes 28.9 (32 datasets in SMALL, 20 in LARGE). We do not have room to tabulate the results of all divisions, but only summarize our findings.

Datasets with a high number of classes (more than 4) produce large, deep trees, and the optimal subtree tends to be very large. Datasets with many examples (more than 1200) display the same characteristics, despite the datasets in our experiments that have more classes having fewer examples, and the datasets having more examples having fewer classes. These groups do not display the same tendency for accuracy. Datasets with more examples produced the most accurate trees, while datasets with more classes produced trees with the lowest accuracy rates.

Missing attributes do not lead to poor classification accuracy. Datasets with missing values had slightly poorer maximum accuracy, but datasets with above average scores had more missing values (and rows with missing values).

These experiments do not show that the best subtree is larger or smaller (percentage-wise) in a large or small tree. The smallest trees with the maximum

	class	egs	%miss rows	%miss vals	nodes	depth	base	AVG	HIGH	FEW	MANY
SMALL	3.44	310.47	20.08	2.83	56.86	9.94	56.54	73.29	83.03	51.88	6.61
LARGE	4.95	502.25	41.24	5.78	70.48	11.23	58.19	76.01	85.12	62.92	5.25

Table 2: Comparing datasets with below and above average numbers of attributes.

score have the roughly the same percentage of nodes of the full tree for both the group of datasets with more than 62 nodes and the group of datasets with 62 or fewer nodes. The group of datasets with trees more than 10 levels deep have larger smallest trees with the maximum score than those the group of datasets with trees that are 10 or fewer levels deep. Still, the group of datasets with smaller optimal trees (less than 55.88% of the nodes in the full tree) average over 13% more nodes than those with larger optimal trees.

We did not feel that we got a lot of information from the "size of the smallest optimal tree" feature. We might have gotten more information by looking at how soon the tree scored within 1% (or 5%, or 10%) of the maximum score. The average score over all nodes was supposed to give data of this nature, but the group with lower average scores reaches their the maximum score earlier than the higher average scores. Average scores are more influenced by the baseline accuracy and the maximum score.

5 CONCLUSIONS

One of the main goals of this investigation has been to improve pruning techniques by identifying optimal orderings in which to generate trees. We considered the possibility of generating decision trees using the PARENT criterion to select attribute splits. We could create new nodes for the attribute value with the maximum number of examples from the parent node's minority class(es). Such a simple counting algorithm stumbles, because it is biased towards frequently occurring, unrelated attribute values, which might be prevalent in examples from both the parent's majority class and the other class(es). Changing the selection to subtract the number of parents's majority class examples (from the number of minority class examples) overcomes this problem. Further examination is left to future studies.

When looking for a fixed point at which to prune a tree, any subtree larger than 42% of the full tree will produce better accuracy than a random prune for the PARENT'S ordering. (We determined the random prune accuracy for a sort with our AVG measure, which scores 77.33%.) The best normalized fixed cut (60% of the full tree) is only slightly less accurate (80.95%) than C4.5's pruning mechanism (82.66%) which is slightly less than optimal (84.62%). All of these are more accurate than the full trees, which classify 80.9% of the examples correctly. Pruned trees averaged 35.75 nodes/dataset, 64.92% the size of the full trees.

The above results and those of the previous section provide guidance for controlling a decision-tree induction method, given the properties of the domain. We hope to uncover more such relationships in this and other forms of inductive learning to reduce overfitting based on knowledge of the method's tendencies in different domains.

Several aspects of the investigation could be improved. The large standard deviations in many of the dataset features suggests a more discretized approach for grouping features. Fuzzy groups, such as few examples, many examples, and enormous number of examples would have normalized the groupings where 10 domains had thousands of examples while the other forty might have dozens. Also, looking at a single binary split of the domain is greatly inferior to the true amount of analysis needed.

References

[Holder, 1995] L. B. Holder. Intermediate decision trees. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1056–1062, 1995.

[Mingers, 1989a] J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243, November 1989.

[Mingers, 1989b] J. Mingers. An empirical comparison of selection measures for decision tree induction. *Machine Learning*, 3(4):319–342, March 1989.

[Quinlan, 1993] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[Schaffer, 1993] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10(2):153–178, 1993.