Comparing Tree-Simplification Procedures

Leonard A. Breslow and David W. Aha
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory
Washington, DC 20375
{breslow,aha}@aic.nrl.navy.mil
http://www.aic.nrl.navy.mil/~ {breslow,aha}

1 Problem and Contribution

Induced decision trees are frequently used by researchers in the machine learning and statistics communities to solve classification tasks (Breiman et al. 1984; Quinlan 1993). However, their practical utility is limited by difficulties users have in comprehending them due to their size and complexity. Many methods have been proposed to simplify decision trees, but their relative capabilities are largely unknown; their evaluation is usually limited to comparisons with "bench-mark" systems (e.g., C4.5, CART). This paper presents a categorization framework for tree-simplification methods and focuses on the empirical comparison of selected methods.

2 Framework and Example Methods

Figure 1 summarizes our framework and lists (in parentheses) the ten example systems we compare in Section 3. Our survey (Breslow & Aha 1997) reviews the research literature on tree simplification within this framework. In this paper, we selected algorithms to compare based on their availability,[1] on their reported performance in previous research, and on our desire to include representatives from each category in the framework. Few of these algorithms have been compared directly with one another.

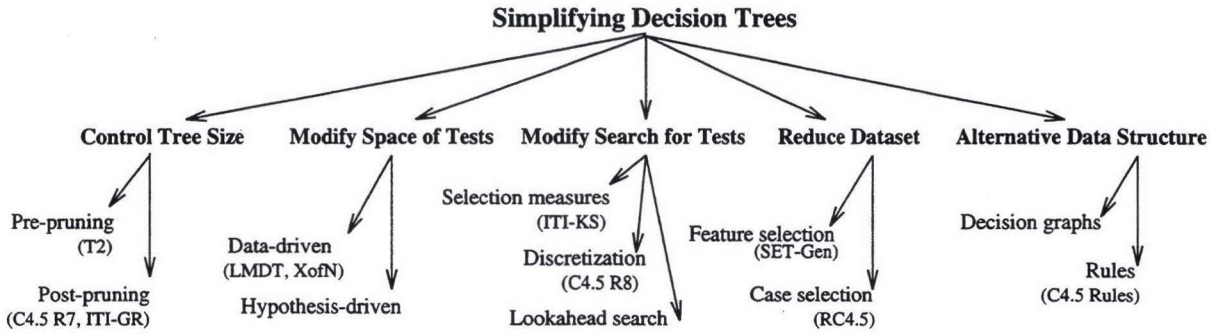

Figure 1: A Framework for Tree-Simplification Procedures

Our framework includes five main categories. The first category consists of pruning algorithms that directly control tree size. *Pre-pruners* terminate tree expansion when the best split fails to attain a threshold on predicted worth. This can result in premature termination, preventing induction from finding valuable combinations of tests. However, we selected T2 (Auer et al. 1995), which limits trees to a depth of 2, because it performed comparatively well and it is an extreme foil for our baseline, C4.5 release 7 (R7) (Quinlan 1993) without pruning. *Post-pruners* completely expand the

[1] In this initial comparison, time constraints prevented us from re-implementing these algorithms and limited us to using existing implementations.

tree, until all leaf nodes are homogeneous with regard to class membership, and then prune. They are intended to enhance noise tolerance, prevent overfitting, and eliminate small disjuncts. We selected C4.5 R7's *error-based post-pruning* (EBP) method, which Esposito et al. (1995) found to yield high accuracy trees, although it tends to underprune. We also selected ITI (ITI-GR) (Utgoff 1995), which, like C4.5, uses gain ratio to select tests, but uses the minimum description length (MDL) principle to guide post-pruning. MDL is an intuitively appealing approach to post-pruning that has not yet been extensively used for tree simplification (see Quinlan & Rivest 1989).

Algorithms in the second category modify the set of possible tests for partitioning cases in a node. Most early algorithms were limited to *univariate* tests, which test the value of one feature. In contrast, *multivariate* algorithms allow tests that reference the values of multiple features. They can be useful for tackling the *subtree replication problem* (Yang et al. 1991), since multivariate tests can cluster cases that univariate tests would separate. Multivariate algorithms often induce trees with fewer nodes than univariate algorithms. However, multivariate tests are more complex, and these algorithms typically have higher computational complexity since they explore a larger space of tests. We selected two promising (data-driven) multivariate algorithms. The first is LMDT (Brodley & Utgoff 1995), which targets numeric features by training perceptrons at each node. LMDT compares favorably with other numeric multivariate approaches (Brodley & Utgoff 1995; Murthy et al. 1994). The second, XOFN (Zheng 1995), targets symbolic features. Its tests consist of an arbitrary set of feature-value pairs. Zheng showed that XOFN reduces tree size and increases accuracy compared with similar approaches.

Methods in the third category modify the search for tests without affecting their representation, either by using alternative test selection measures, discretizing continuous data, or automating lookahead. We selected a variant of ITI, named ITI-KS, that differs from ITI-GR only in its use of Kolmogorov-Smirnoff distance rather than gain ratio to select tests. Utgoff and Clouse (1996) found that it consistently produced smaller trees without sacrificing accuracy. We also selected C4.5 R8 (Quinlan 1996), which discretizes continuous features before induction. It uses an MDL approach to penalize continuous features having many values, and often increases accuracy and reduces tree size relative to C4.5 R7.

Methods in the fourth category reduce the database, either through *feature selection* or *case selection*. Both have been shown to dramatically reduce tree size. We selected one example from each subcategory. SET-GEN (Cherkauer & Shavlik 1996) performs feature selection, using a genetic algorithm to search the space of feature subsets. In contrast, Robust C4.5 (RC4.5) (John 1995) selects cases by iteratively inducing and pruning a tree using C4.5 R7. In each iteration it discards cases that are misclassified. It terminates when a new tree correctly classifies all remaining cases. Both algorithms performed well compared with C4.5 R7, but, like most of the algorithms we tested, have not been compared with one another.

Algorithms in the final category transform a decision tree into another data structure. We selected C4.5 RULES (Quinlan 1993), a simplification method that transforms a tree into a set of rules, which are then pruned. Pérez and Rendell (1995) reported that C4.5 RULES yields more accurate predictors than the C4.5 tree-induction algorithm.

3 Experiment

Our goal here is not to determine which approach for simplifying decision trees is "best" because different learning approaches are preferable under different conditions (Schaffer 1993). Some of these approaches are certainly complementary (e.g., Robust C4.5 and SET-GEN can be used to pre-process a case base for several of the other algorithms). Others are limited to performing well

Table 1: Data Set Characteristics

name	#cases	#feat	#class	%num	%symb	miss	%most freq.
autos	205	26	7	62	38	yes	37
breast cancer-w	198	30	2	100	0	yes	76
credit	690	15	2	40	60	yes	56
diabetes-pi	768	8	2	100	0	no	65
heart-disease	920	13	5	50	50	yes	45
kr-vs-kp	3196	36	2	0	100	no	52
promoters	106	58	2	0	100	no	50
tic-tac-toe	958	9	2	0	100	no	65

Table 2: Selected Datasets: + and − Indicate Good and Poor Comparative Error (E) and Size (S) Results

Dataset	Algorithms															
	T2		LMDT		XofN		ITI-KS		C4.5 R8		SET-Gen		RC4.5		Rules	
	E	S	E	S	E	S	E	S	E	S	E	S	E	S	E	S
autos									+		-					
breast (W)			+	-								+		+		
credit		+							+	+	+	+				
diabetes	-	+				-	+	+	+	+			+	+	+	
heart (Cl)	-		+		+	+	+	+	+	+		+		+		
KRKP	-															-
promoter	-				+	+	-					+			+	
tic-tac-toe					+	+							-	-	+	

under certain sets of conditions (e.g., T2 performs best for tasks where trees with low depth are preferable). However, it is not clear how simplification methods compare across categories of our framework. The purpose of this study is to gain insight into their relative capabilities. Towards this goal, this section describes our initial empirical comparisons of tree simplification algorithms. We discuss future research goals in Section 4.

3.1 Initial Experiment

We selected eight datasets (Table 1) from the UCI Machine Learning Database Repository (Merz 1996) for our experiments. For each dataset, at least one of the selected algorithms performed either particularly well or poorly compared with C4.5 in previous studies (e.g., SET-Gen's greatest simplifications occurred on the credit dataset, C4.5 R8's largest gains in accuracy occurred with the autos dataset). Table 2 summarizes some of these previous results. We trained and tested each algorithm using the same set of folds in a ten-fold cross validation study. Compared with the other algorithms, LMDT was at a disadvantage because it requires a pruning set; thus we used seven folds for training with LMDT (as opposed to nine folds for the other algorithms) and two for pruning. The ITI algorithms were run in batch, rather than incremental, mode. We describe the parameter settings used when testing these algorithms in Breslow and Aha (1996).

Figures 2 and 3 respectively summarize average error rate ratios (i.e., the algorithm's average error relative to the baseline's average error) and percent tree size (relative to the baseline), where the baseline results were obtained using C4.5 R7 without pruning. The dashed lines group the algorithms by their category in our framework. We defined the size of univariate trees as their

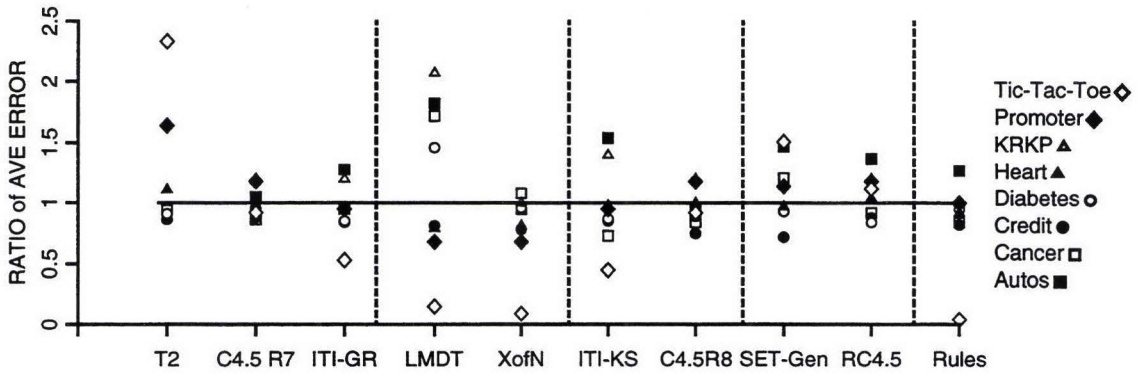

Figure 2: Ratios of Average 10-Fold CV Error versus C4.5 R7 *without* Pruning

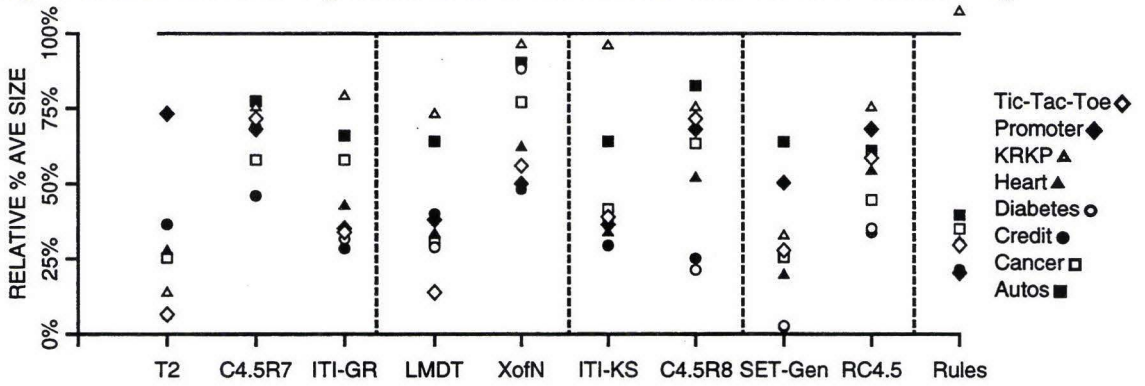

Figure 3: Average 10-Fold CV Tree Sizes versus C4.5 R7 *without* Pruning

number of nodes. For multivariate trees, we summed the number of features used at internal nodes with the number of leaves. For C4.5 RULES, we counted the number of rule conditions. Not shown in the figures are two error ratios for the KRKP dataset: T2's error ratio was 26.2 and SET-GEN's was 5.76. Also, T2 could not be tested on the autos dataset because it is computationally expensive to apply it to classification tasks with more than four classes.

Turning first to the accuracy findings, it was not surprising to find that the C4.5 "family" of algorithms (i.e., C4.5 R7, C4.5 R8, ROBUSTC4.5, C4.5 RULES) displayed accuracies most similar to the baseline algorithm, C4.5 R7 without pruning. XOFN was generally superior to the baseline, while T2 was generally inferior. The two ITI algorithms' results were variable, as were those for LMDT[2] and SET-GEN.

The results on size were generally more variable than those on accuracy. Not surprisingly, the algorithms were more consistently superior to the baseline in size than in accuracy. Some trade-offs between size and accuracy appeared. For instance, T2 traded accuracy for (small) size, while XOFN traded size for accuracy. Similarly, LMDT and SET-GEN produced small trees, but had highly variable accuracies. Algorithms displaying both good accuracy and size were C4.5 R8, ROBUSTC4.5, and especially C4.5 RULES. Although we did not record exact execution speeds,[3] we found that LMDT, XOFN, and SET-GEN were very slow.

Some interactions between the algorithms' performance and the datasets are noteworthy. The multivariate algorithms both performed well on tic-tac-toe, whose target concept is easily described by carefully chosen feature combinations. As expected, C4.5 R8 improved performance relative to

[2]LMDT's accuracy was also highly variable across some of the datasets' folds (e.g., its standard deviation was 21.7% for the breast cancer task).

[3]Due to the size of these experiments, we tested different algorithms on different CPUs, which complicated our attempt to compare their speeds.

Table 3: Parameters tested for the C4.5 Variants (R7, R8, Robust) (values used in initial experiment are in *italics*)

Parameter	Notation	Values Tested
Pruning confidence level	-c x	{1,5,10,15,*25*,35,45}
Attribute grouping	-s	{grouping, *no grouping*}
Stopping criterion/weight	-m x	{1,*2*,3,5,10,15}

Table 4: Parameters for C4.5 RULES (values used in initial experiment are in *italics*)

Parameter	Notation	Values Tested
Attribute redundancy ratio	-r x	{.9,*1.0*,1.1,1.3,1.5,2.0,2.5}
Rule condition confidence level	-F x	{No testing,.01,.05,.10,.15,*.25*}
Pruning confidence level	-c x	{1,3,5,10,15,*25*,35,45}

R7 on datasets with numeric features (e.g., diabetes). Finally, C4.5 RULES performed well, especially on tasks where rules are appropriate (e.g., tic-tac-toe), but actually increased data structure size on the KRKP task, which requires a complex rule description. It also had low accuracy on the autos dataset, which has more classes than the other datasets and thus requires more rules.

The dataset reduction algorithms reduced tree sizes, but their accuracies were disappointing, probably because most of these datasets are highly preprocessed and do not benefit from reduction. When tested on a proprietary dataset with nearly 200 features, most of which are irrelevant, SET-GEN excelled. This is not surprising because it was designed to tolerate irrelevant features.

3.2 Results with parameter tuning

The results described in Section 3.1 used fixed values for the parameters of the selected algorithms. Thus, these results might not be good indicators of their comparative performance when their parameters are carefully tuned. We therefore replicated the previous experiment, including parameter tuning where possible.

Parameter tuning was only possible for algorithms whose implementations include adjustable parameters and that are efficient enough for exploring a reasonably-sized space of parameter values. Only the four algorithms in the C4.5 family met these criteria. These algorithms were tested under the same conditions as described in Section 3.1, except that algorithm parameters were automatically tuned for each fold. Parameter tuning focused on maximizing classification accuracy only. Tuning was performed on a per-fold basis using the same ten folds as in the preceding experiment. Specifically, we exhaustively tested all combinations of the parameter values listed in Tables 3 and 4 using a nine-fold cross validation on each of the ten original folds. The selected parameter settings were then tested on the remaining fold that was not involved in the parameter tuning.

The results for accuracy and size are shown in Figures 4 and 5, respectively. The baseline is the same as in the preceding experiment, C4.5 R7 without pruning and *without parameter tuning.* Parameter tuning yields improvements in both accuracy and size. However, for some datasets (e.g., promoter and autos) and for C4.5 RULES the improvements were generally small or negligible. In contrast, for the tic-tac-toe dataset, accuracy improvements were particularly strong. Interestingly, parameter tuning generally benefited size more than accuracy, even though tuning was guided by accuracy only.

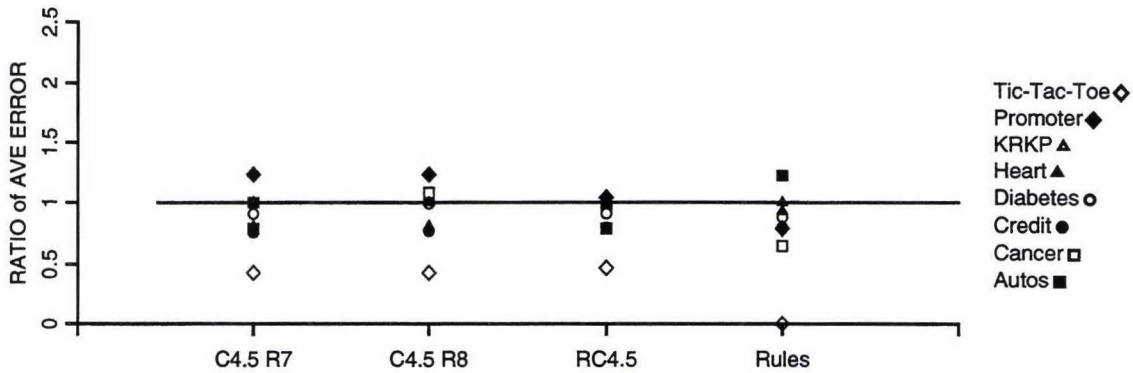

Figure 4: Ratios of Average 10-Fold CV Error versus C4.5 R7 *without* Pruning, with Parameter Tuning

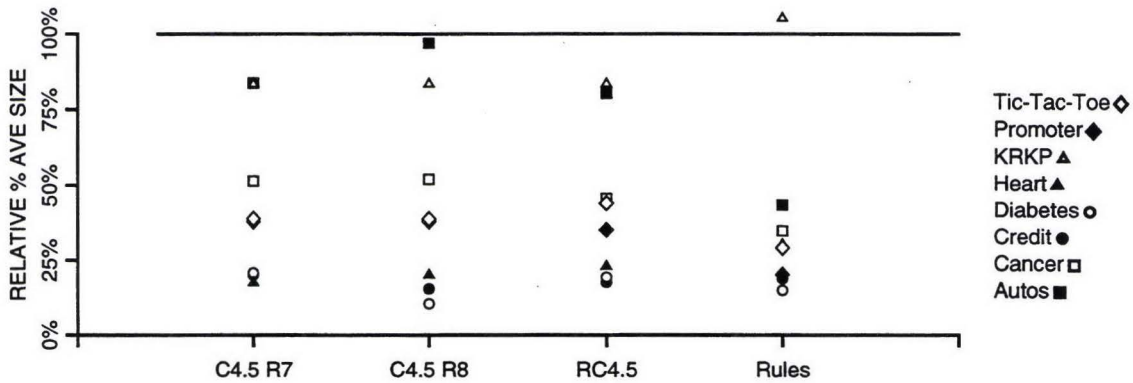

Figure 5: Average Ten-Fold CV Tree Sizes versus C4.5 R7 *without* Pruning, with Parameter Tuning

4 Discussion

Considering both accuracy and size, we found that the improved versions of C4.5 (i.e., C4.5 R8, ROBUSTC4.5, and C4.5 RULES) generally performed best. These were also among the more efficient algorithms tested and have tunable parameters. Parameter tuning yields improvements in size or accuracy in some cases.

This paper describes an initial empirical comparison of tree simplification methods. Our ultimate goal is to determine which ones are best for supporting case-retrieval in a state-of-the-art tool for case-based reasoning (Aamodt & Plaza 1994). While decision trees have been used case retrieval mechanisms, tree simplification methods have not been studied in this context. Simplification methods have been primarily limited to classification tasks; we have not yet tested extensions of them for case retrieval. Our future research on case retrieval will focus on testing a larger suite of tree simplification approaches for both carefully constructed and benchmark problem-solving tasks, where the evaluation measures will include retrieval precision and recall rather than classification accuracy. Additional measures will include the appropriateness of retrieved cases for subsequent combination, adaptation, and problem solving.

References

Aamodt, A. and E. Plaza (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, *7*, 39–59.

Auer, P., R. C. Holte, and W. Maass (1995). Theory and applications of agnostic PAC-learning with small decision trees. In *Proceedings of the Twelfth International Conference on Machine*

Learning (pp. 21–29). Tahoe City, CA: Morgan Kaufmann.

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

Breslow, L. and D. W. Aha (1996). *Tree-simplification procedures for case retrieval* (Technical Report AIC-96-14). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

Breslow, L. A. and D. W. Aha (1997). Simplifying Decision Trees: A Survey. To appear in *Knowledge Engineering Review.*

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

Cherkauer, K. J. and J. W. Shavlik (1996). Growing simpler decision trees to facilitate knowledge discovery. In *Proceedings of the Knowledge Discovery and Data Mining Conference.* Portland, OR: AAAI Press.

Esposito, F., D. Malerba, and G. Semeraro (1995). A further study of pruning methods in decision tree induction. *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics.* (pp. 211–218). Ft. Lauderdale, FL.

John, G. (1995). Robust decision trees: Removing outliers in databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining.* Montreal, Canada: AAAI Press.

Merz, C. (1996). *UCI Repository of machine learning databases* [Machine-readable data repository @ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

Murthy, S., S. Kasif, and S. Salzberg (1994). A system for the induction of oblique decision trees. *Journal of Artificial Intelligence Research, 2,* 1–32.

Pérez, E. and L. A. Rendell (1995). Using multidimensional projection to find relations. In *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 447–455). Tahoe City, CA: Morgan Kaufmann.

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

Quinlan, J. R. (1996). Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research, 4,* 77–90.

Quinlan, J. R. and R. L. Rivest (1989). Inferring decision trees using the minimum description length principle. *Information and Computation, 80,* 227–248.

Schaffer, C. (1993). Overfitting avoidance as bias. *Machine Learning, 10,* 113–152.

Utgoff, P. E. (1995). *Decision tree induction based on efficient tree restructuring* (Technical Report 95-18). Amherst, MA: University of Massachusetts, Department of Computer Science.

Utgoff, P. E. and J. A. Clouse (1996). *A Kolmogorov-Smirnoff metric for decision tree induction* (Technical Report 96-3). Amherst, Massachusetts: University of Massachusetts, Department of Computer Science.

Yang, D., G. Blix, and L. A. Rendell (1991). The replication problem: A constructive induction approach. In *Proceedings of the European Working Session on Learning* (pp. 44–61). Porto, Portugal: Springer-Verlag.

Zheng, Z. (1995). Constructing nominal X-of-N attributes. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1064–1070). Montreal: Morgan Kaufmann.