

SE-trees Outperform Decision Trees in Noisy Domains

Ron Rymon*

Intelligent Systems Program, 901 CL
University of Pittsburgh
Pittsburgh, PA 15260
Rymon@ISP.Pitt.edu

Abstract

As a classifier, a Set Enumeration (SE) tree can be viewed as a generalization of decision trees. It can be shown that, at the cost of a higher complexity, a single SE-tree encapsulates many alternative decision tree structures. An SE-tree enjoys several advantages over decision trees: it allows for domain-based user-specified bias, it supports a flexible tradeoff between the resources allocated to learning and the resulting accuracy, and it can combine knowledge induced from examples with other knowledge sources. In this paper, we empirically demonstrate that SE-trees enjoy a particular advantage over simple decision trees in noisy domains. This advantage manifests itself both in terms of accuracy, and in terms of consistency.

Introduction

Seminal work by Breiman *et al.* (1984) and Quinlan (1986), on induction of decision tree classifiers, was followed by a wealth of research by many others to improve upon the basic method. In (Rymon, 1993), we propose a generalization of this framework using Set Enumeration (SE) trees. In an SE-tree, every node can possibly be expanded with *multiple* attributes; Systematic enumeration ensures that rules are uniquely represented. SE-trees enjoy several advantages over simple decision trees, including the possible use of domain-based user-specified bias, and the ability to flexibly trade off resources allocated to learning (time, space) to get better models. An SE-tree can be shown to embed many alternative decision trees, allowing for explicit resolution of conflicts whenever they exist. Close relationship between an SE-tree's rules and prime implicants allows a natural way to combine knowledge induced from the examples with other knowledge sources (Rymon, 1994).

SE-trees do, however, involve additional complexity. Although this additional complexity can be controlled within a hill-climbing procedure, users shall always be concerned whether the extra effort will pay off. This paper is an attempt to partly address this question.

*Parts of this work were supported by NASA grant MTPE-94-02; and funding from Modeling Labs.

Specifically, we investigate the effect of *noise* on the marginal improvement offered by the SE-tree framework. Since noise is prevalent in any real-world domain, this is an important question for data mining practitioners as well.

We take an empirical approach. Controlling for other structural parameters, we randomly generate classification problem sets with varying noise levels. For each problem, we first explore a simple decision tree, and then expand a complete SE-tree. Using 30 different problem sets for each reported data point, we record average accuracies and complexities, as well as variances. We conclude that SE-trees are generally advantageous in noisy domains, and that for moderate noise levels this advantage increases with additional noise.

Section reviews SE-tree-based induction; the interested reader is referred to (Rymon, 1993; Rymon, 1994) for more in-depth presentations. Section outlines the hypothesis and experimental framework. Section presents experimental results, and Section concludes with a discussion of the results and related work.

SE-tree-based Induction

Set Enumeration trees were first proposed as a systematic way to search a space of sets (Rymon, 1992). As a framework for induction, SE-trees generalize decision trees in that a node can be expanded according to *multiple* attributes.

At every node, candidate attributes are first scored according to the chosen attribute-selection heuristic (e.g. information gain), and then expansion proceeds in that order. Uniqueness of exploration is guaranteed using the concept of a node's *View*. The root's *View* consists of all attributes. Another node's *View* consists of all attributes in its parent's *View* which scored higher than its own. Consider the SE-tree depicted in Figure 1. Without loss of generality, suppose that attributes first scored in a lexicographic order, as marked next to the root. The root was thus expanded with all attribute-value pairs, in that order.

Next, in every node, we re-score attributes in that node's *View* according to same heuristic. For example,

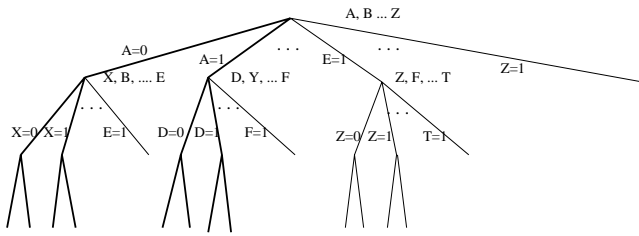


Figure 1: Improving Upon a Given Decision Tree

in the node $\{E=1\}$, attributes F-Z are re-ranked: Z, F, ... T; in the node $\{Z=1\}$, the *View* is empty. Now note that, at the left side of the SE-tree, one finds the nodes that would have appeared in a decision tree utilizing same heuristic (arcs of this *primary decision tree*, appear in bold).

Unfortunately, the complete SE-tree may often be too large to be entirely searched. Nodes are explored according to a user-specified *exploration policy*. One particular family of exploration policies begins with nodes of the primary decision tree, and only then continues to explore other parts of the SE-tree, e.g., as in a beam search, or by cardinality, remaining entropy, etc. The extent of exploration can depend on the resources available and the importance of a more accurate model. Note that the exploration policy is a form of user-specified bias, in the learning phase.

Search is generally aimed at *kernel* (minimal) rules. When evaluated, each node is either determined to be a kernel rule, and is thus kept as a leaf; determined not to lead to a kernel rule, and is thus dismissed; or expanded with attributes in its *View*. When a new kernel rule is discovered, all previously discovered rules which are subsumed by it are discarded. Due to its multi-attribute branching, an SE-tree can also be viewed as a *union* of many alternative decision trees. It is, however, an economical representation of this union: all overlaps are uniquely represented, and only most general rules are retained.

The SE-tree's redundancy plays a role in the classification phase as well. Whereas in a decision tree, a given instance can only match a single path, multiple matching paths may exist in an SE-tree. Furthermore, for instances not in the training set, these alternative decision rules may disagree in their classification. These conflicts introduce another opportunity for a user-specified bias, in the form of a *resolution criterion*. A resolution criterion can be simple, e.g., simple voting, or it may represent some domain knowledge, e.g., one attribute can represent more dependable measurements than another.

This family of algorithms for SE-tree-based induction, including a variety of attribute-selection measures, as well as exploration policies and resolution criteria, is implemented in a program called SE-Learn (SE-Learn Home Page).

The Hypothesis and Method of Investigation

We hypothesize that the SE-tree outperforms decision trees in noisy domains. As noted, the SE-tree embeds many alternative decision tree structures. In regions of the domain where all these classifiers agree, the SE-tree will simply echo this agreement. However, where there is a disagreement, its resolution criterion mechanism allows consideration of the evidential support of the opposing views. Even if conflicts are resolved via a simple voting mechanism, it still has the smoothing effect of an averaging process. In the presence of noise, some induced rules may be contaminated. In the decision tree framework, all instances matched by a contaminated rule will necessarily be misclassified. In contrast, in the SE-tree framework, the influence of contaminated rules is mitigated by that of clean rules during conflict resolution.

We investigate our hypothesis empirically, using randomly generated artificial problem sets. (We chose to use artificial data sets because it is easier to control their parameters, and to see that they cover a range of problem types.) In each experiment, we generate induction problems with a varying noise level, and a fixed set of other parameters. For each problem, we test *generalization accuracy* of both the SE-tree and its primary decision tree. Note that this is enough because an SE-tree can be generated around every chosen decision tree. We control for the following parameters:

- **Problem size:** 10 binary attributes, and a binary class. These are small enough to allow experimentation with many problem sets, yet are large enough to be meaningful.
- **Training set size:** 25% of the domain. We also report experiments with 5% and 50% of the domain used for training.
- **Form of target function:** Disjunctive Normal Form (DNF). Every discrete function can be represented that way.
- **Complexity of target function:** diversity is achieved by varying the number of conjunctions, and the number of literals per conjunction. It is hard to syntactically characterize the complexity of a DNF function's complexity. Instead, we chose to focus on two parameters which we believe are significant in determining a problem's complexity: the number of conjunctions ($\#\text{conj}$) in the function, and the number of literals mentioned in each conjunction ($\#\text{lits}$).
- **Distribution of problems:** Uniform over a given choice of $\#\text{conj}$ and $\#\text{lits}$.
- **Choice of decision tree program:** in most experiments, we used Information Gain (ID3) to select splitting attributes. We obtained concurring results, however, using GINI-Index (CART), Gain Ratio (C4.5), and Chi-Square (ChAID).

- **Parametrization of the SE-tree framework.**
 - **Exploration policy and extent:** we used complete SE-trees. Complete SE-trees are typically *less accurate* classifiers than some partially explored sub-trees, making this a conservative assumption. Also, this gets around the problem of choosing a particular exploration policy.
 - **Resolution criterion:** simple voting. Again a conservative assumption.
- **Pruning:** we used unpruned trees in most experiments. We report verifying experiments where both the decision tree and SE-tree were subjected to significance-based statistical pruning.

Results

We randomly generate artificial problem sets. In each experiment, we fix a choice of values for the generation parameters, while varying the noise level from 0% to 100%. For each data point, to account for random variability, we generate 30 different problems. For each problem, unless noted otherwise, 25% of the domain instances are randomly chosen to be included in the training set, with the rest used for testing. Noise levels are modeled as the percentage of randomly-chosen training instances which are labeled with a random class. Test set instances are all labeled with their correct class. For each problem, we use the training set to first generate a primary decision tree, and then a complete SE-tree. We record the generalization accuracy, as well as variance and complexity, of both classifiers.

Specifically, for each data point, we measure

1. average error rate for the decision tree and SE-tree – $\text{Err}(\text{DT})$ and $\text{Err}(\text{SE})$ respectively;
2. standard deviation of these error rates – $\text{STD}(\text{DT})$ and $\text{STD}(\text{SE})$; and
3. number of rules in each of the classifiers – $\text{Size}(\text{DT})$ and $\text{Size}(\text{SE})$.

We then compute:

1. the normalized reduction in error rates when expanding from the decision tree to the SE-tree: $(\text{Err}(\text{DT}) - \text{Err}(\text{SE})) / \text{Err}(\text{SE})$;
2. the normalized reduction in variance: $(\text{STD}(\text{DT}) - \text{STD}(\text{SE})) / \text{STD}(\text{SE})$; and
3. the ratio between the complexities of the two classifiers: $\text{Size}(\text{SE}) / \text{Size}(\text{DT})$.

(In all graphs, the first two measures are multiplied by 100, to convey percentage terms.)

In our first set of experiments, we held $\#\text{lits}$ constant at 5, and varied $\#\text{conj}$ from 5 to 20 to 50. This varies the shape and complexity of the target function. In general, 20-conjunction functions are *a priori* more complex than the other two. (Ignoring noise, 5- and 50-conjunction problems consist of mostly negative and positive examples respectively, compared to

about even distribution for 20-conjunction functions; 5-conjunction functions are also less fragmented.)

Figure 2(a) presents the percent-improvement in error rates gained by expanding from the simple decision tree to the complete SE-tree. We first note that, in general, error rates are decreased using the SE-tree; this is demonstrated by most of the values being positive. We also note that the relative accuracy first increases as more noise is introduced, and then drops back until there is no improvement at all. This demonstrates that, at moderate noise levels, the SE-tree is more resilient to noise, i.e., that its accuracy deteriorates at a slower pace than the decision tree’s. Then, however, as more noise is introduced, the training set contains less and less information, and so error rates must converge to 50% for *all* non-oblivious classifiers at the 100% level. Also, as absolute error rates increase with higher noise levels, it becomes harder to score high in percent-terms reductions.

Figure 2(b) presents the percent reduction in variance gained when expanding from the simple decision tree to the complete SE-tree. We note that SE-tree performs more consistently for moderate noise levels. With higher noise levels, it becomes less consistent.

Finally, Figure 2(c) presents complexity ratios. First, we note that (as expected) the complete SE-tree is substantially larger than the simple decision tree. Second, we note that ratios runs within a very narrow range (13 to 15.5). Finally, we note that ratios are typically higher for more complex functions. However, we also note that increasing noise seems to induce a decrease in the ratio for the more complex functions (and an increase for simpler functions). Our intuitive explanation is that, when there is little or no noise, the SE-tree’s complexity is necessary to account for the function complexity whereas the decision tree is too simple. Then, as more noise is added, it introduces new complexities. In the SE-tree, however, part of this new complexity cancels out with the function complexity.

In our second set of experiments, we held $\#\text{conj}$ constant at 20, and varied $\#\text{lits}$ from 3 to 5 to 7. This too varies the shape and complexity of the target function. In general, for reasons similar to those explained above, 5-conjunct functions are *a priori* more complex than the other two. Results obtained in these experiments (Figure 3) were very similar to those reported for the previous experiment. Here too, error rate reduction is lower for the more complex functions.

Following these experiments, we proceeded to investigate the effect of three choices we made in our experimental design: the choice of information gain as a representative heuristic; the use of 25% of the domain for training; and the decision not to use pruning. To remain as conservative as possible, we chose to use the combination of parameters for which the relative performance of the SE-tree was the poorest, i.e., $\#\text{conj}=20$ and $\#\text{lits}=5$.

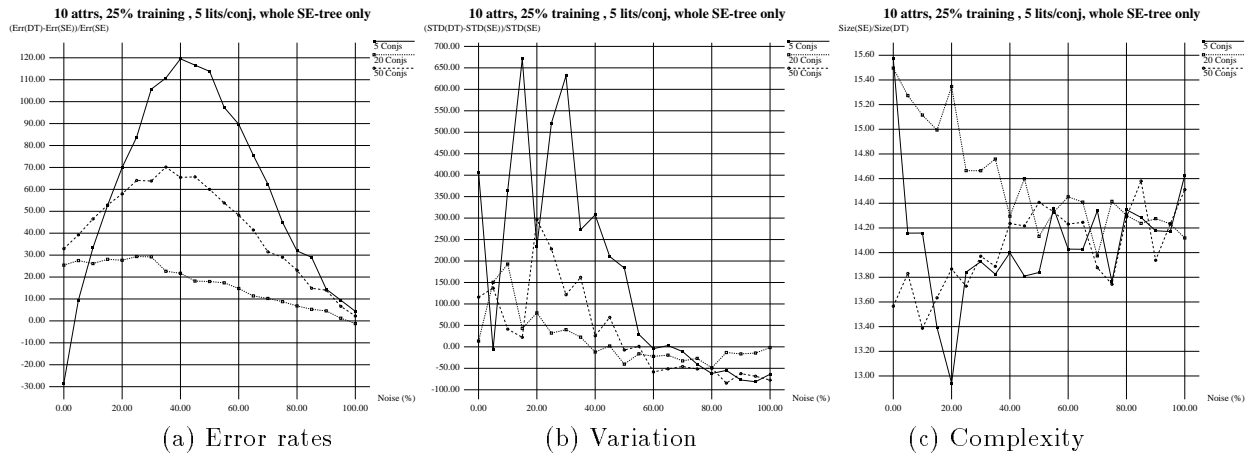


Figure 2: Varying #conj

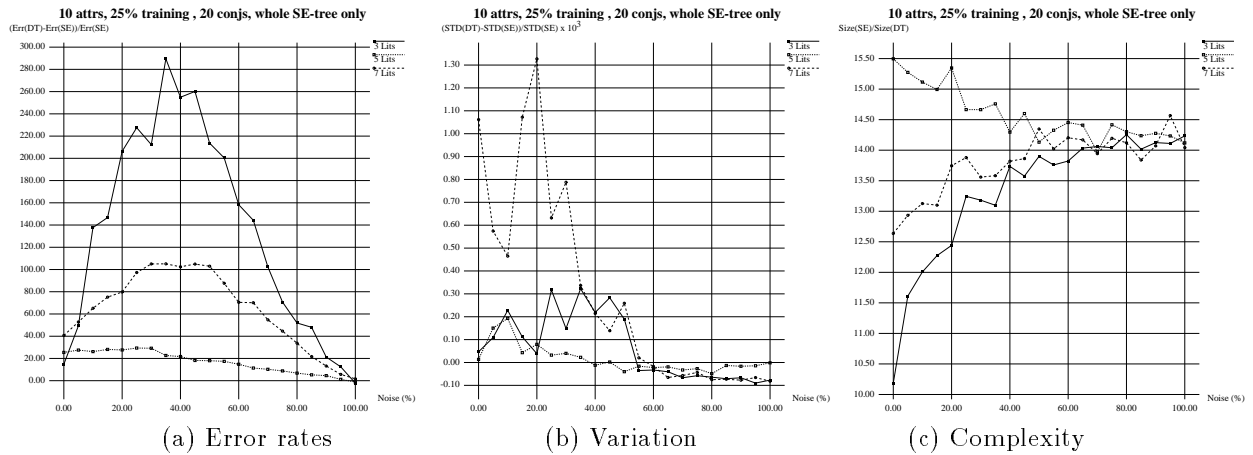


Figure 3: Varying #lits

In a first set of confirming experiments, we tested three other attribute-selection heuristics: Gain Ratio as used in C4.5 (Quinlan, 1993), GINI Index as used in CART (Breiman *et al.*, 1984), and Chi Square as used in ChAID. As Figure 4 shows, we found very little differences in the performances of alternative heuristics.

In a second set of confirming experiments, we tested alternative training set sizes. We tested generalization accuracy using 5%, 25%, and 50% of the domain for training (Figure 5). Generally, results are similar to what we have encountered with 25% training sets. We do notice, however, that percent reduction in error rates are lower when fewer instances are available for training. However, much of this is because the absolute error rates are much higher then. We also notice that complexity ratios are higher when fewer instances are used for training.

Finally, in a third set of confirming experiments, we tested the effect of pruning. We used statistical pruning because this is the only form or pruning implemented in SE-Learn. Specifically, we used bino-

mial significance testing to prune the primary decision tree at the $p < 0.05$ level, and to add rules from other parts of the SE-tree with same significance level. Figure 6(a) shows that while the SE-tree still generally outperforms the decision tree, the differences are smaller when pruning is used. Figure 6(b) shows that the variance itself varies substantially more when pruning is used. Figure 6(c) shows that complexity ratios are generally smaller, i.e., SE-trees are much closer to decision trees in their sizes, when pruning is used. More interestingly, when pruning is used, the ratio declines sharply as additional noise is introduced.

Discussion and Conclusion

Results of our experiments demonstrate that:

1. In the presence of noise, a complete SE-tree typically outperforms its own primary decision tree. While it does not follow that an SE-tree outperforms any decision tree, an SE-tree can be constructed around any *specific* decision tree. Plus, we have replicated

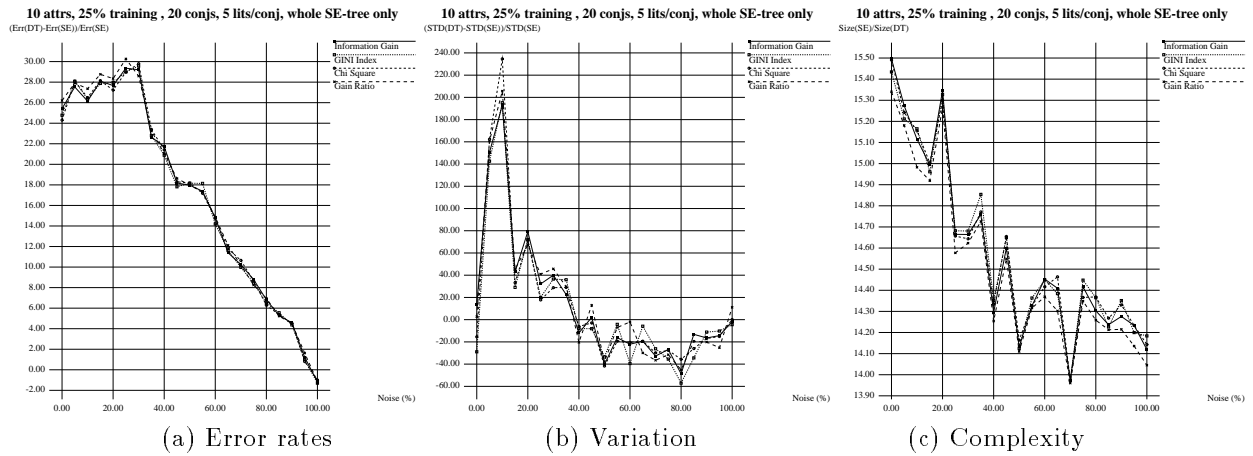


Figure 4: Various Attribute-Selection Heuristics

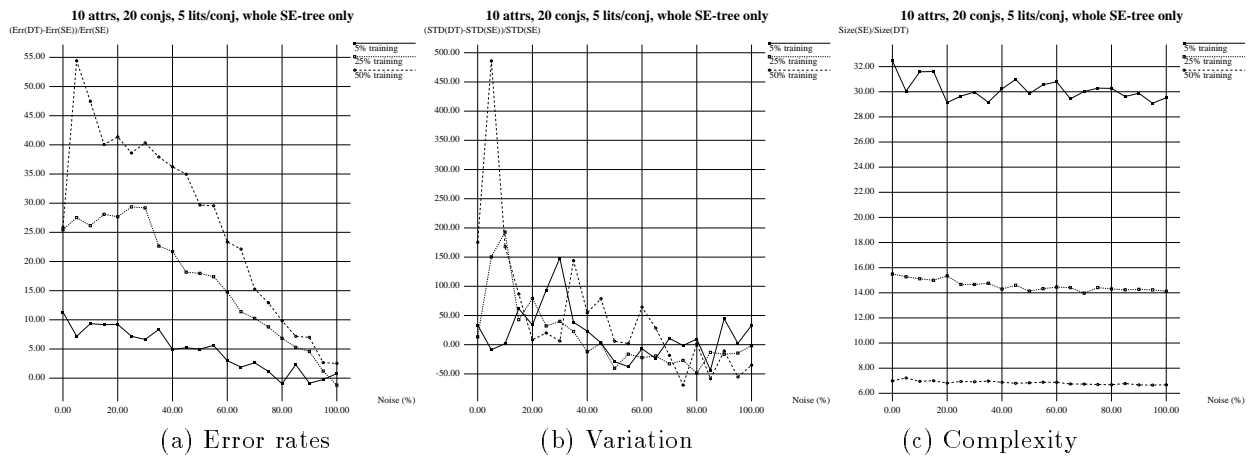


Figure 5: Various Training Set Sizes

our results with four of the more common decision tree construction schemes. Also note that our results are conservative in that we limited experiments to complete SE-trees, despite the fact that partially explored SE-trees are often better performers.

2. Where noise levels are moderate, the SE-tree advantage generally *increases* with additional noise. Later, noise overcomes information content in the training data, and the accuracy of both classifiers drop to the level of random classifiers at the 100% noise level.
3. Where noise levels are moderate, applied to problems with similar structural characteristics, the SE-tree is also more consistent.

Considering that the SE-tree encompasses many alternative decision trees, the SE-tree-based framework bears resemblance to the tree averaging approach taken by Kwok and Carter (1990) and Buntine (1994), as well as to Breiman’s bagging (1994).

The explanation to the SE-tree’s superior performance may well lie in its reduced variance. In analyzing

the reasons why bagging works, Breiman notes that bagging is effective where there is great variance between alternative predictors. In the bagging approach, alternative predictors are each constructed using a different subset of the training data. As described, bagging could also be applied to SE-trees. However, even the plain SE-tree framework already pools together many alternative decision trees, each corresponding to an alternative construction heuristic. Note also that this is accomplished without the loss of information incurred when subsetting the training data.

When noise is first added to a training set, it corrupts the signature of what is presumably the correct structure in the data. This expands the space of candidate hypotheses, and hence increases in the diversity of the decision trees embedded within the SE-tree. Consequently, the predicted performance of a single decision tree drawn from that population becomes more volatile, making the SE-tree a more attractive option.

Dietterich and Kong (1995) distinguish two components in an algorithm’s error rate: statistical bias, and

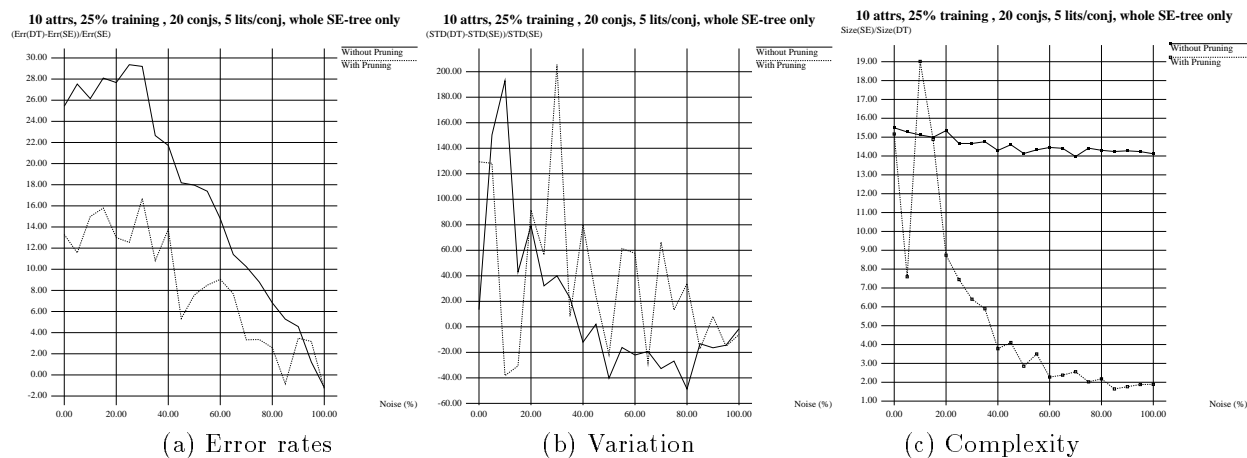


Figure 6: With and Without Pruning

variance. Statistical bias is the error of the average predictor. The variance is among alternative predictors. To reduce the error rate, one shall therefore reduce either or both. In their experiments, Dietterich and Kong show that variance can account for a significant part of misclassifications by the C4.5 algorithm. They even go as far as indicating that “one important source of variance in C4.5 is the fact that the algorithm must choose a single split at each node in the tree” (p. 7). The SE-tree’s multiple splits, together with its embedding of many alternative decision trees act to reduce the variance error component. Also, recall that for moderate noise levels the SE-tree enjoyed a lower variance than its primary decision tree. While this variance is computed over many problem sets with same features, and not over different training sets for the same problem as in (Breiman, 1994; Dietterich & Kong, 1995), the two are clearly closely related. Finally, having experimented with tree averaging and error-correcting procedures, Dietterich and Kong conclude that “Some method is needed for converting a combination of trees (or other complex hypotheses) into a smaller, equivalent hypothesis. These trees are very redundant; how can we remove this redundancy, while still reducing bias and variance?”. In our opinion, SE-trees go one clear step in that direction.

Acknowledgements

I thank Kevin Atteson, Ron Kohavi, Greg Provan, Foster Provost, Philip Resnik, Cullen Schaffer, Bob Schrag, and members of the Pitt ML group for their helpful comments and suggestions on earlier drafts.

References

Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees*. Wadsworth, Belmont.

Breiman, L., Bagging Predictors. *Technical Report 421*, Department of Statistics, UC Berkeley.

Buntine, W., Learning Classification Trees. *Artificial Intelligence Frontiers in Statistics*, D. Hand (Ed), Chapman and Hall.

Dietterich, T. G., and Kong, E. B., Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms. Technical Report, Department of Computer Science, Oregon State University.

Kwok, S., and Carter, C., Multiple Decision Trees. *Uncertainty in Artificial Intelligence*, 4, pp. 327-335.

Quinlan, J. R., Induction of Decision Trees. *Machine Learning*, 1(1):81-106.

Quinlan, J. R., *C4.5: Programs for Empirical Learning*. Morgan Kaufmann, San Francisco, CA.

Rymon, R., Search through Systematic Set Enumeration. In Proceedings *Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge MA, pp. 539-550.

Rymon, R., An SE-tree-based Characterization of the Induction Problem. In Proceedings *Tenth International Conference on Machine Learning*, pp. 268-275, Amherst MA.

Rymon, R., On Kernel Rules and Prime Implicants. In Proceedings *Twelfth National Conference on Artificial Intelligence*, Seattle WA, pp. 181-186.

<http://www.isp.pitt.edu/~rymon/SE-Learn.html>.